Relative Positional Encoding for Transformers with Linear Complexity

Antoine Liutkus, Ondřej Cífka, Shih-Lun Wu, Umut Şimşekli, Yi-Hsuan Yang, Gael Richard

Relative Positional Encoding for Transformers with Linear Complexity

Antoine Liutkus [*1] **Ondřej Cífka** [*2] **Shih-Lun Wu** [3 4 5] **Umut Şimşekli** [6] **Yi-Hsuan Yang** [3 5] **Gaël Richard** [2]

Abstract

Recent advances in Transformer models allow for unprecedented sequence lengths, due to linear space and time complexity. In the meantime, relative positional encoding (RPE) was proposed as beneficial for classical Transformers and consists in exploiting lags instead of absolute positions for inference. Still, RPE is not available for the recent linear-variants of the Transformer, because it requires the explicit computation of the attention matrix, which is precisely what is avoided by such methods. In this paper, we bridge this gap and present *Stochastic Positional Encoding* as a way to generate PE that can be used as a replacement to the classical additive (sinusoidal) PE and provably behaves like RPE. The main theoretical contribution is to make a connection between positional encoding and cross-covariance structures of correlated Gaussian processes. We illustrate the performance of our approach on the Long-Range Arena benchmark and on music generation.

1. Introduction

1.1. Linear Complexity Transformers

The Transformer model (Vaswani et al., 2017) is a new kind of neural network that quickly became state-of-the-art in many application domains, including the processing of natural language (He et al., 2020), images (Dosovitskiy et al., 2020), audio (Huang et al., 2018; Pham et al., 2020) or bioinformatics (AlQuraishi, 2019) to mention just a few.

The core, novel component of the Transformer is the *attention layer*. It computes M output values $\mathbf{y}_m$ from N input values $\mathbf{v}_n$, all being vectors of an arbitrary dimension. Following classical non-parametric regression principles (Nadaraya, 1964; Watson, 1964), it consists in a simple weighted sum:

$$\mathbf{y}_m = \frac{\sum_n a_{mn}\mathbf{v}_n}{\sum_n a_{mn}}\,, \tag{1}$$

where each *attention* coefficient $a_{mn} \in \mathbb{R}_+$ – gathered in the $M \times N$ matrix $\mathbf{A}$ – indicates how important the value $\mathbf{v}_n$ is in the computation of the output $\mathbf{y}_m$.

One of the main contributions of the Transformer is an original method to compute these coefficients. D-dimensional feature vectors $\mathbf{k}_n$ and $\mathbf{q}_m$ are attached to all items of the input and output sequences and are called *keys* and *queries*, respectively. Gathering them in the $N \times D$ and $M \times D$ matrices $\mathbf{K}$ and $\mathbf{Q}$, we get *softmax dot-product attention* as:

$$\mathbf{A} = \exp\left(\mathbf{Q}\mathbf{K}^\top \Big/ \sqrt{D}\right) \equiv [a_{mn} = \mathcal{K}(\mathbf{q}_m, \mathbf{k}_n)]_{mn}\,, \tag{2}$$

where the function exp is applied element-wise. The right-hand side in (2) is a generalization introduced by Tsai et al. (2019) and Choromanski et al. (2020), where $\mathcal{K}$ is a *kernel* function. Parameters pertain to how keys $\mathbf{k}_n$, values $\mathbf{v}_n$ and queries $\mathbf{q}_m$ are obtained from the raw sequences, usually by time-distributed fully connected layers.

The original Transformer architecture (Vaswani et al., 2017) explicitly computes the attention matrix $\mathbf{A}$, leading to a $\mathcal{O}(MN)$ complexity that prevents it from scaling to very long sequence lengths. Although this is not necessarily a problem when sequence lengths are barely on the order of a few hundreds, as in some language processing tasks, it is

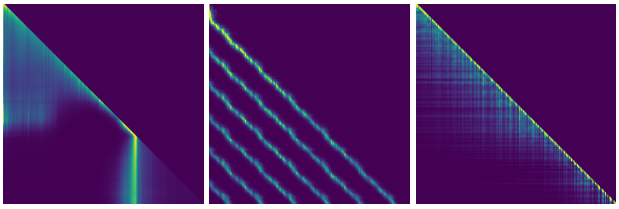


Figure 1. Examples of attention patterns observed in the Performers trained for pop piano music generation (section 3.2) at inference time, for sequence length $M = N = 3\,072$ while training sequences have length $2\,048$. (left) Absolute PE. (middle) Sinusoidal SPE. (right) Convolutional SPE. Note that SPE never requires computing these full attention patterns.

[*]Equal contribution [1]Inria, Zenith Team, UMR LIRMM, Univ. Montpellier, France [2]LTCI, Télécom Paris, Institut Polytechnique de Paris, France [3]Research Center for IT Innovation, Academia Sinica, Taiwan [4]National Taiwan University, Taiwan [5]Taiwan AI Labs, Taiwan [6]INRIA – Département d'Informatique de l'École Normale Supérieure – PSL Research University, Paris, France. Correspondence to: Liutkus Antoine <antoine.liutkus@inria.fr>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

prohibitive for very large signals like high-resolution images or audio.

Focusing on this scalability issue, several approaches have been recently investigated to allow for long sequences:

- *Attention clustering* schemes group items among which dependencies are computed through regular attention. This is either done by using simple proximity rules within the sequences, leading to chunking strategies (Dai et al., 2019), or by clustering the keys and values (Roy et al., 2020). Inter-cluster dependencies are either ignored or summarized via fixed-length context vectors that are coined in as *memory* (Wu et al., 2020).
- Assuming the attention matrix to be *sparse*. In this case, only a few a_{mn} are nonzero (Child et al., 2019).
- Assuming $\mathbf{A}$ has a particular (low-rank) *structure* and can be decomposed as the product of two smaller matrices. A prototypical example is the Linformer (Wang et al., 2020b), which is limited to fixed-length inputs. Another very recent line of research in this same vein takes:

$$\mathbf{A} \approx \phi(\mathbf{Q})\phi(\mathbf{K})^\top, \tag{3}$$

  where $\phi : \mathbb{R}^D \to \mathbb{R}^R$ is a non-linear *feature map* applied to each key $\mathbf{k}_n$ and query $\mathbf{q}_m$, and $R \ll \min(M, N)$ (Shen et al., 2020; Katharopoulos et al., 2020).
- When $\mathcal{K}$ in (2) is a positive (semi)definite kernel, the Performer (Choromanski et al., 2020) leverages *reproducing kernel Hilbert spaces* to show that a random ϕ may be used to exploit this convenient decomposition (3) *on average*, even when $\mathbf{A}$ is not low rank:

$$\mathcal{K} \succeq 0 \Leftrightarrow \mathbf{A} = \mathbb{E}_\phi \left[\phi(\mathbf{Q})\phi(\mathbf{K})^\top\right], \tag{4}$$

  where ϕ is drawn from a distribution that depends on $\mathcal{K}$. A simple example is $\phi_{\mathbf{W}}(\mathbf{k}_n) = \max(0, \mathbf{W}\mathbf{k}_n)$, with a random $\mathbf{W} \in \mathbb{R}^{R\times D}$ for some $R \in \mathbb{N}$.

Whenever an efficient scheme like (3) or (4) is used, the outputs can be obtained without computing the attention coefficients a_{mn}, as in (10).[1]

1.2. Positional Encoding

In Transformer networks, the outputs $\mathbf{y}_m$ are computed as linear combinations of *all* input values $\mathbf{v}_n$, weighted by attention coefficients a_{mn}. In sequence modeling, it is reasonable to assume that the actual *positions* m and n should play a role in the computation, in addition to the *content* at these locations; otherwise, any permutation of the sequence would lead to the same output. Two core approaches were undertaken to incorporate position information:

- The original Transformer (Vaswani et al., 2017) adds this information to the inputs of the network, i.e. before the first attention layer. This can be equivalently understood as augmenting the keys, values and queries:

$$\mathbf{k}_n \leftarrow \mathbf{k}_n + \bar{\mathbf{k}}_n,\ \mathbf{v}_n \leftarrow \mathbf{v}_n + \bar{\mathbf{v}}_n,\ \mathbf{q}_m \leftarrow \mathbf{q}_m + \bar{\mathbf{q}}_m, \tag{5}$$

  where we write $\bar{\mathbf{k}}_n \in \mathbb{R}^D$ for the *keys positional encoding* (PE; Sukhbaatar et al., 2015) at position $n \in \mathbb{N}$ and analogously for values and queries. Vaswani et al. propose a deterministic scheme based on trigonometric functions, which is shown to work as well as trainable embeddings.
- As an example of positional encoding *in the attention domain*, a *relative positional* encoding (RPE) was proposed by Shaw et al. (2018), building on the idea that time lags $m - n$ are more important than absolute positional encoding (APE) for prediction. It is written as:

$$\mathbf{A} = \exp\Big(\big(\mathbf{Q}\mathbf{K}^\top + \mathbf{\Omega}\big)\Big/\sqrt{D}\Big), \text{ with:} \tag{6}$$

$$\mathbf{\Omega} \equiv \left[\omega_{mn} = \sum_{d=1}^{D} q_{md}\mathcal{P}_d(m-n)\right]_{mn}. \tag{7}$$

  The terms $\mathcal{P}_d$ now act as D different encodings for *time lags* selected based on the queries. This change is advocated as bringing important performance gains in many application areas and has enjoyed a widespread use ever since.

Although writing down the positional encoding in the attention domain is beneficial for performance (Shaw et al., 2018; Dai et al., 2019; Tsai et al., 2019), we are only aware of implementations that either require the computation of $\mathbf{A}$, or clustered attention schemes, which *in fine* decompose $\mathbf{A}$ into smaller attention matrices, and *compute them*. This is in sharp contrast to (3) and (4), which never compute the attention matrix.

Our contributions can be summarized as follows:

- We propose *Stochastic Positional Encoding* (SPE) as a general PE scheme *in the keys domain*, that enforces a particular attention pattern devised *in the attention domain*. This enables RPE without explicit computation of attention. To our knowledge, it is the first RPE strategy that is compatible with $\mathcal{O}(N)$ Transformers like Choromanski et al. (2020) and Katharopoulos et al. (2020).
- We study the impact of SPE on performance on the Long-Range Arena benchmark (Tay et al., 2021) and two music generation tasks. Since RPE was so far limited to short sequences, we believe this is the first study of its advantages on long-range predictions. Our results demonstrate better validation losses and extrapolation ability.
- We provide additional resources on our companion website,[2] including Python implementations of SPE for PyTorch and JAX/Flax.

[1] A somewhat related strategy is used by the recent LambdaNetworks (Bello, 2020), which encapsulate the key-value information as a so-called *lambda* function to be applied query-wise, hence also avoiding the computation of a full attention matrix.

[2] https://cifkao.github.io/spe/

Algorithm 1 Stochastic Positional Encoding.

Input

- position kernel $\mathcal{P}(m,n)$, number of replicas R.
- initial $M \times D$ and $N \times D$ queries $\mathbf{Q}$ and keys $\mathbf{K}$.

Positional encoding:

- Draw the D independent couples $\{\overline{\mathbf{Q}}_d, \overline{\mathbf{K}}_d\}_d$ of $M \times R$ and $N \times R$ matrices as in section 2.1
- Set $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{K}}$ as in (16) and (17)

Inference compute outputs $\mathbf{Y}$ with the $\mathcal{O}(N)$ Transformer:

$$\mathbf{Y} \leftarrow \mathrm{diag}(\mathbf{d})^{-1}\left[\phi\Big(\widehat{\mathbf{Q}}\Big)\left[\phi\Big(\widehat{\mathbf{K}}\Big)^\top \mathbf{V}\right]\right] \tag{10}$$

with $\mathbf{d} = \phi(\widehat{\mathbf{Q}})\left[\phi\Big(\widehat{\mathbf{K}}\Big)^\top \mathbf{1}_N\right]$ and ϕ discussed in (3)/(4).

2. Stochastic Positional Encoding

Index set and notation. We assume that the input/output sequences are indexed by $n, m \in \mathbb{T}$, where $\mathbb{T}$ is the *index set*. For regularly sampled sequences, we have $\mathbb{T} = \mathbb{N}$, but more settings are possible, like irregularly sampled time series ($\mathbb{T} = \mathbb{R}$) or images ($\mathbb{T} = \mathbb{N}^2$). In any case, the particular lists of input / output locations under consideration are written: $\mathcal{N}$ and $\mathcal{M}$, with respective sizes N and M (the case $\mathcal{N} = \mathcal{M}$ is called *self-attention*). The corresponding keys and values are hence indexed as $\{\mathbf{k}_n\}_{n\in\mathcal{N}}$ and $\{\mathbf{v}_n\}_{n\in\mathcal{N}}$, while queries are $\{\mathbf{q}_m\}_{m\in\mathcal{M}}$. For convenience, we write a_{mn} for the entries of the $M \times N$ attention matrix $\mathbf{A}$.
We use bold uppercase for matrices, bold lowercase for vectors and a NumPy-like notation: if $\mathbf{X}_k$ is a $I \times J$ matrix, $\mathbf{x}_{k,i}$ and $\mathbf{x}_{k,:,j}$ stand for its i^{th} row and j^{th} column, respectively.

Assumptions. In the remainder of this paper, we will seek an attention matrix $\mathbf{A}$ given by:

$$\mathbf{A} = \exp\left(\left[\sum_{d=1}^{D} q_{md}\mathcal{P}_d(m,n)k_{nd}\right]_{mn} \Big/ \sqrt{D}\right), \tag{8}$$

where $\{\mathcal{P}_d\}_{d=1}^D$ are *position kernels*. Defining $\mathbf{P}_d \equiv [\mathcal{P}_d(m,n)]_{mn}$, this can be written in matrix form as:

$$\mathbf{A} = \exp\left(\sum_{d=1}^{D} \mathrm{diag}\big(\mathbf{q}_{:,d}\big)\mathbf{P}_d\,\mathrm{diag}(\mathbf{k}_{:,d}) \Big/ \sqrt{D}\right), \tag{9}$$

which is understood as having D positional attention templates $\mathbf{P}_d$ jointly activated by the queries $\mathbf{q}_{:,d}$ and keys $\mathbf{k}_{:,d}$. Original RPE (7) can be seen as a special case, where some entries are kept constant.

Positional attention as covariance. The key idea for SPE is to see the attention kernel $\mathcal{P}_d(m,n)$ as a *covariance*:

$$(\forall \mathcal{M}, \mathcal{N})\,(\forall m, n)\,\mathcal{P}_d(m,n) = \mathbb{E}\left[\overline{Q}_d(m)\overline{K}_d(n)\right], \tag{11}$$

where $\overline{Q}_d(m)$ and $\overline{K}_d(n)$ are two real and zero-mean random variables, which will be chosen with the single condition that their covariance function matches $\mathcal{P}_d$. Semantically, they should be understood as (randomly) encoding position m for queries and position n for keys, respectively. When multiplied together as in dot-product attention, they yield the desired attention template $\mathcal{P}_d(m,n)$ on average. The central intuition is that the actual positional encodings do not matter as much as their dot-product.

In what follows, we will impose specific structures on the cross-covariance $\mathcal{P}_d(m,n)$, which will in turn allow us to design *random processes* $\overline{Q}_d = \{\overline{Q}_d(m)\}_{m\in\mathcal{M}}$ and $\overline{K}_d = \{\overline{K}_d(n)\}_{n\in\mathcal{N}}$ such that (11) holds. The core advantage of this construction is to allow for $\mathbf{P}_d$ to be factorized. Let us for now assume that we construct the distributions of $\{\overline{Q}_d(m), \overline{K}_d(n)\}_d$ in such a way that we can sample from them (we will see how in section 2.1) and consider R independent realizations of them for given $\mathcal{M}$ and $\mathcal{N}$, gathered in the $M \times R$ and $N \times R$ matrices $\overline{\mathbf{Q}}_d$ and $\overline{\mathbf{K}}_d$:

$$\overline{\mathbf{Q}}_d \equiv [q_{d,m,r} \sim \overline{Q}_d(m)]_{mr}, \quad \overline{\mathbf{K}}_d \equiv [k_{d,n,r} \sim \overline{K}_d(n)]_{nr}. \tag{12}$$

For large R, by the law of large numbers, we obtain:

$$\mathbf{P}_d \approx \left[\overline{\mathbf{Q}}_d\overline{\mathbf{K}}_d^\top\right]/R. \tag{13}$$

This leads $\mathbf{A}$ in (9) to be given by:

$$\mathbf{A} \approx \exp\left(\sum_{d=1}^{D} \mathrm{diag}\big(\mathbf{q}_{:,d}\big)\frac{\overline{\mathbf{Q}}_d\overline{\mathbf{K}}_d^\top}{R}\,\mathrm{diag}(\mathbf{k}_{:,d})/\sqrt{D}\right) \tag{14}$$

$$\approx \exp\frac{\left(\sum_{d=1}^{D} \mathrm{diag}\big(\mathbf{q}_{:,d}\big)\overline{\mathbf{Q}}_d\right)\left(\sum_{d=1}^{D} \mathrm{diag}(\mathbf{k}_{:,d})\overline{\mathbf{K}}_d\right)^\top}{R\sqrt{D}}. \tag{15}$$

Here, a *crucial* observation is that for large R, the cross-terms $\overline{\mathbf{Q}}_d\overline{\mathbf{K}}_{d'\neq d}^\top$ are negligible due to independence, provided that the means of the processes are selected to be zero. Finally, picking queries and keys as:

$$\widehat{\mathbf{Q}} \leftarrow \sum_{d=1}^{D} \mathrm{diag}\big(\mathbf{q}_{:,d}\big)\overline{\mathbf{Q}}_d/\sqrt[4]{DR}\,, \tag{16}$$

$$\widehat{\mathbf{K}} \leftarrow \sum_{d=1}^{D} \mathrm{diag}(\mathbf{k}_{:,d})\overline{\mathbf{K}}_d/\sqrt[4]{DR}\,, \tag{17}$$

we see from (15-17) that we get back to the usual multiplicative scheme (2) with $\mathbf{A} = \exp(\widehat{\mathbf{Q}}\widehat{\mathbf{K}}^\top/\sqrt{R})$, where the queries/keys now have dimension R and can be used in (10) to directly get outputs without computing $\mathbf{A}$.

The procedure is summarized in Algorithm 1: we provide a way (16-17) to achieve PE in the *keys domain*, such that the desired model (8) is enforced in the *attention domain*, pa-

rameterized by the attention kernels $\mathcal{P}_d$. Interestingly, this is done without ever computing attention matrices, complying with $\mathcal{O}(N)$ Transformers. The remaining challenge, which we discuss next, is to generate $\overline{\mathbf{Q}}_d$ and $\overline{\mathbf{K}}_d$ enforcing (13).

2.1. Drawing Stochastic Positional Encodings

Inspecting (11), we notice that our objective is to draw samples from D pairs of centered random processes $\{\overline{Q}_d, \overline{K}_d\}_d$, with a prescribed cross-covariance structure $\mathcal{P}_d$. It is reasonable to use Gaussian processes for this purpose (Williams & Rasmussen, 2006), which have the maximum entropy for known mean and covariance. Such distributions are frequently encountered in geophysics in the *co-kriging* literature (Matheron, 1963; Genton & Kleiber, 2015), where scientists routinely handle correlated random fields. The particular twists of our setup are: we have a *generative* problem, e.g. as in Vořechovský (2008); however, as opposed to their setting, we are not directly interested in the marginal covariance function of each output, provided that the desired cross-covariance structure holds.

The most straightforward application of SPE arises when we pick $\mathcal{P}_d(m,n) = \mathcal{P}_d(m-n)$, i.e. a stationary position kernel, which was coined in as choosing *relative* attention in Shaw et al. (2018) and boils down to enforcing a *Toeplitz* structure for the cross-covariance matrix $\mathbf{P}_d \equiv [\mathcal{P}_d(m-n)]_{m,n}$ between $\overline{Q}_d$ and $\overline{K}_d$.

We propose two variants of SPE to handle this important special case, illustrated in Figure 2. The first variant yields *periodic* covariance functions. It can be beneficial whenever attention should not vanish with large lags, as in traffic prediction (Xue & Salim, 2020) or, as we show, in music generation. The second variant generates *vanishing* covariance functions; a concept which has recently been shown useful (Wang et al., 2021), and notably yields smaller validation losses in some of our experiments.

Variant I. Relative and periodic attention (`sineSPE`). In our first approach, we consider the case where $\mathcal{P}_d$ is periodic, which gets a convenient treatment. We assume:

$$\mathcal{P}_d(m,n) = \sum_{k=1}^{K} \lambda_{kd}^2 \cos(2\pi f_{kd}\,(m-n) + \theta_{kd})\,, \quad (18)$$

where $K \in \mathbb{N}$ is the number of *sinusoidal* components and $\mathbf{f}_d \in [0\ 1]^K$, $\boldsymbol{\theta}_d \in [-\pi\ \pi]^K$ and $\boldsymbol{\lambda}_d \in \mathbb{R}^K$ gather their K frequencies, phases, and weights, respectively. By using the matrix notation, we can rewrite (18) as:

$$\mathbf{P}_d = \boldsymbol{\Omega}(\mathcal{M}, \boldsymbol{f}_d, \boldsymbol{\theta}_d)\,\mathrm{diag}\Big(\ddot{\boldsymbol{\lambda}}_d\Big)^2\,\boldsymbol{\Omega}(\mathcal{N}, \boldsymbol{f}_d, \mathbf{0})^\top, \quad (19)$$

where $\ddot{\mathbf{v}} \equiv \big[v_{\lfloor p/2 \rfloor}\big]_p \in \mathbb{R}^{2K}$ denotes a twice upsampled version of a vector $\mathbf{v} \in \mathbb{R}^K$, $\lfloor\cdot\rfloor$ denotes the floor operation, and for an index set $\mathcal{I}$, $\boldsymbol{\Omega}(\mathcal{I}, \mathbf{a}, \mathbf{b})$ is a matrix of size $|\mathcal{I}| \times 2K$, with entries (0-based indexing):

$$[\boldsymbol{\Omega}\,(\mathcal{I}, \boldsymbol{a}, \boldsymbol{b})]_{nl} = \begin{cases} \cos(2\pi a_k n + b_k) & \text{if } l = 2k \\ \sin(2\pi a_k n + b_k) & \text{if } l = 2k+1 \end{cases}$$

It can be shown that if $\boldsymbol{\theta}_d = \mathbf{0}$ and $\mathcal{M} = \mathcal{N}$, we get back to the (unique) Vandermonde decomposition for positive definite Toeplitz matrices[3] (Yang et al., 2016), which boils down in our context to assuming that $\forall \tau, \mathcal{P}_d(0) \geq \mathcal{P}_d(\tau)$. Since this is not always desirable, we keep the more general (19).

At this point, we can easily build $\overline{\mathbf{Q}}_d$ and $\overline{\mathbf{K}}_d$. We draw a $2K \times R$ matrix $\mathbf{Z}_d$ with independent and identically distributed (i.i.d.) Gaussian entries of unit variance, and define:

$$\overline{\mathbf{Q}}_d \leftarrow \boldsymbol{\Omega}(\mathcal{M}, \boldsymbol{f}_d, \boldsymbol{\theta}_d)\,\mathrm{diag}\Big(\ddot{\boldsymbol{\lambda}}_d\Big)\mathbf{Z}_d/\sqrt{2K}\,, \quad (20)$$

$$\overline{\mathbf{K}}_d \leftarrow \boldsymbol{\Omega}(\mathcal{N}, \boldsymbol{f}_d, \mathbf{0})\,\mathrm{diag}\Big(\ddot{\boldsymbol{\lambda}}_d\Big)\mathbf{Z}_d/\sqrt{2K}\,. \quad (21)$$

It is easy to check that such a construction leads to (13). Its parameters are $\{\mathbf{f}_d, \boldsymbol{\theta}_d, \boldsymbol{\Lambda}_d\}_d$, which can be trained through stochastic gradient descent (SGD) as usual.

Variant II. Relative (vanishing) attention with regular sampling (`convSPE`). Due to their periodic structure, the covariance functions generated by Variant I are *non-vanishing*. Yet, our framework is flexible enough to allow for vanishing covariance structures, which may be more desirable depending on the application (Wang et al., 2021).

As opposed to Variant I, where we imposed a specific structure on $\mathcal{P}_d$, we will now follow an indirect approach, where $\mathcal{P}_d$ will be *implicitly* defined based on our algorithmic construction. In this case, we assume that the signals are regularly sampled (typical in e.g. text, images, audio), and we will exploit the structure of Gaussian random matrices and basic properties of the convolution operation.

For ease of notation, we assume self attention, i.e. $\mathcal{M} = \mathcal{N}$. Let $\{\boldsymbol{\Phi}_{\boldsymbol{d}}^{\boldsymbol{Q}}, \boldsymbol{\Phi}_d^K\}_d$ denote a collection of *filters*, which will ultimately be learned from training data. The size and the dimension of these filters can be chosen according to the input data (i.e. can be vectors, matrices, tensors). We then propose the following procedure, which leads to a Toeplitz $\mathbf{P}_d$ by means of *convolutions*:

- We first draw an $M \times R$ random matrix $\mathbf{Z}_d$ with i.i.d. standard Gaussian entries. For multidimensional signals, $\mathbf{Z}_d$ gathers R random vectors, matrices, cubes, etc.
- The desired $\overline{\mathbf{Q}}_d$ and $\overline{\mathbf{K}}_d$ are obtained by convolving $\mathbf{Z}_d$ with respective filters $\boldsymbol{\Phi}_{\boldsymbol{d}}^{\boldsymbol{Q}}$ and $\boldsymbol{\Phi}_d^K$:

$$\overline{\mathbf{Q}}_d = \mathbf{Z}_d * \boldsymbol{\Phi}_d^Q\,,\ \overline{\mathbf{K}}_d = \mathbf{Z}_d * \boldsymbol{\Phi}_d^K\,, \quad (22)$$

  where $*$ denotes convolution with appropriate dimension (e.g. 1D, 2D or 3D). Using convolutions with finite filters

[3] If $\mathbf{P}_d \succeq 0$ and $K \geq N$, (19) still holds but is not unique.

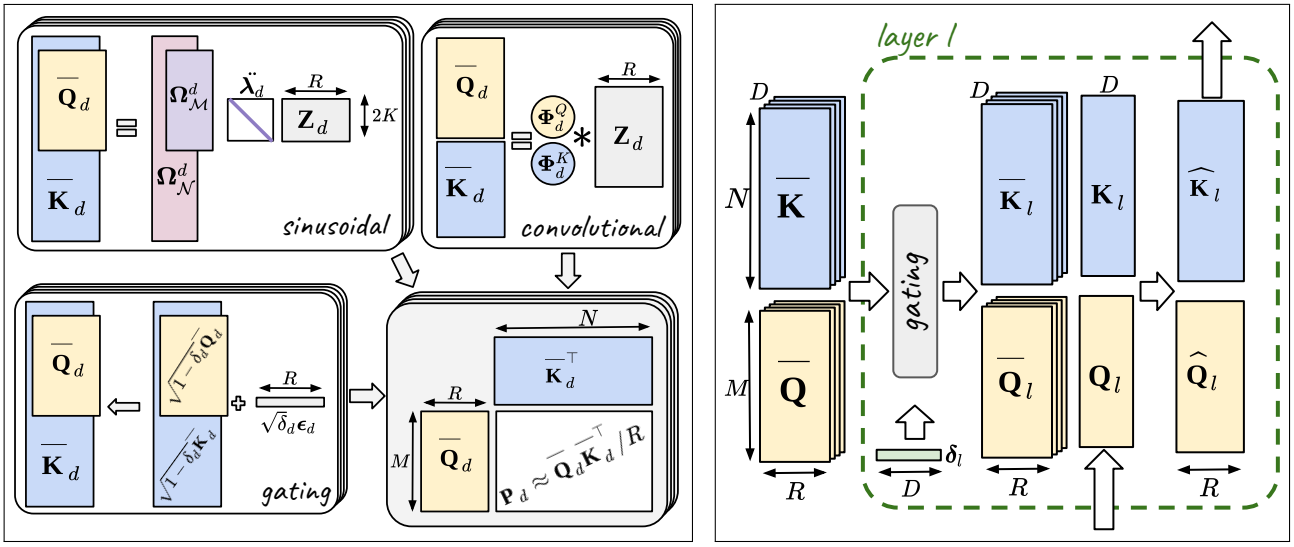


Figure 2. (left) Generation of $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ in SPE, which approximate the templates $\mathbf{P}_d$ when multiplied together. (right) $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ can be shared across layers. At each layer l, different gating is (optionally) used, before applying (16-17) to generate new queries $\mathbf{Q}$ and keys $\mathbf{K}$.

ensures vanishing covariance, as proven in the appendix.

Due to the independence of the entries of $\mathbf{Z}_d$, for large R, the product $\mathbf{Z}_d\mathbf{Z}_d^\top/R$ will tend to the identity matrix. Given the fact the convolution operations in (22) can be equivalently expressed as a multiplication by triangular Toeplitz matrices constructed from the respective filters, it can be shown that, as $R \to \infty$, $\frac{1}{R}\overline{\mathbf{Q}}_d\overline{\mathbf{K}}_d^\top$ tends to the product of two triangular Toeplitz matrices. Hence, by using the properties of triangular Toeplitz matrices (cf. Kucerovsky et al. 2016, Theorem 2.4), we conclude that, as $R \to \infty$, our construction yields a Toeplitz matrix $\mathbf{P}_d$ as desired. This approach is parameterized by the filters$\{\mathbf{\Phi}_d^Q, \mathbf{\Phi}_d^K\}_d$, which will be learned from training data through SGD.

The variety of attention patterns $\mathcal{P}(m-n)$ that can be obtained directly depends on the kernel sizes, which is a classical result from signal processing (Vetterli et al., 2014). Cascading several convolutions as in the VGGNet (Simonyan & Zisserman, 2014) may be a convenient way to augment the expressive power of this convolutional SPE variant.

From a more general perspective, the two operations in (22) can be understood as producing PE through filtering white noise, which is the core idea we introduce for PE. Other classical signal processing techniques may be used like using *infinite impulse response* filters. Such considerations are close to the ideas proposed in (Engel et al., 2020).

To summarize, the core difference between the two proposed constructions (20-21) and (22) lies in the behaviour of RPE beyond a maximum lag, implicitly defined through the frequencies $\mathbf{f}_d$ for (20-21) and through the sizes of the filters for (22). While the sinusoidal construction leads to a periodic RPE, the filtering construction leads to a vanishing RPE, which is called *monotonic* in (Wang et al., 2021). Both may be the desired option depending on the application.

2.2. Gated SPE

Although RPE and the generalization (9) we propose are novel and efficient strategies to handle position information, it may be beneficial to also allow for attention coefficients that are computed without positional considerations, simply through $\langle \mathbf{q}_m, \mathbf{k}_n \rangle$. As a general *gating* mechanism, we propose to weight between positional and non-positional attention through a *gate parameter* $\delta_d \in [0\ 1]$:

$$\mathbf{P}_d \equiv [\delta_d + (1-\delta_d)\mathcal{P}_d(m,n)]_{m,n}\,. \tag{23}$$

This gating scheme can be implemented simply by augmenting $\overline{\mathbf{Q}}_d$ and $\overline{\mathbf{K}}_d$ generated as above through:

$$\overline{\mathbf{q}}_{d,m} \leftarrow \sqrt{1-\delta_d}\,\overline{\mathbf{q}}_{d,m} + \sqrt{\delta_d}\boldsymbol{\epsilon}_d\,, \tag{24}$$

$$\overline{\mathbf{k}}_{d,m} \leftarrow \sqrt{1-\delta_d}\,\overline{\mathbf{k}}_{d,m} + \sqrt{\delta_d}\boldsymbol{\epsilon}_d\,, \tag{25}$$

where $\boldsymbol{\epsilon}_d \in \mathbb{R}^R$ in (24) and (25) is the same and has i.i.d. standard Gaussian entries.

In practice, we can share some SPE parameters across the network, notably across layers, to strongly reduce computing time and memory usage. In our implementation, *sharing* means generating a single instance of $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ for each head, on which a layer-wise gating is applied, before achieving PE through (16-17). This is illustrated in Figure 2.

Table 1. Long-Range Arena results (higher scores are better). Mean and standard deviation of accuracy over three runs is reported, except for Performer with convolutional SPE, where only a single run was completed. For comparison, the best result reported by Tay et al. (2021), along with the name of the best-performing model (in parentheses), is included.

	ListOps	Text	Retrieval	Image
Best result from Tay et al. (2021)	37.27 (Reformer)	65.90 (Linear Trans.)	59.59 (Sparse Trans.)	44.24 (Sparse Trans.)
Linear Transformer-ReLU from Tay et al.	18.01	65.40	53.82	42.77
Performer-softmax (`APE`)	**17.80** $\pm$ 0.00	62.58 $\pm$ 0.22	59.84 $\pm$ 1.46	41.81 $\pm$ 1.16
Performer-softmax + `sineSPE`	17.43 $\pm$ 0.32	62.60 $\pm$ 0.50	60.00 $\pm$ 1.20	41.12 $\pm$ 1.70
Performer-softmax + `convSPE`	**17.80**	60.94	57.22	40.06
Linear Transformer-ReLU (`APE`)	17.58 $\pm$ 1.01	63.98 $\pm$ 0.05	58.78 $\pm$ 0.93	**42.25** $\pm$ 0.01
Linear Transformer-ReLU + `sineSPE`	**17.80** $\pm$ 0.00	**64.09** $\pm$ 0.62	**62.39** $\pm$ 0.59	41.21 $\pm$ 1.18
Linear Transformer-ReLU + `convSPE`	9.50 $\pm$ 1.17	63.23 $\pm$ 1.31	61.00 $\pm$ 1.34	39.96 $\pm$ 1.31

3. Experiments

3.1. Long-Range Arena

Experimental setup. We evaluate the proposed method in the Long-Range Arena (LRA; Tay et al., 2021), a benchmark for efficient Transformers, consisting of sequence classification tasks with a focus on long-range dependencies. We use the following tasks from this benchmark:

- *ListOps*: parsing and evaluation of hierarchical expressions. a longer variant of (Nangia & Bowman, 2018);
- *Text*: movie review sentiment analysis on the IMDB corpus (Maas et al., 2011);
- *Retrieval*: article similarity classification on the All About NLP (AAN) corpus (Radev et al., 2013);
- *Image*: object recognition on the CIFAR10 dataset (Krizhevsky, 2009) represented as pixel sequences.

The tasks are challenging due to the large sequence lengths, deliberately increased by choosing a character-/pixel-level representation. An overview of the tasks can be found in the appendix. We do not include *Pathfinder* (a synthetic image classification task) as we were unable to reproduce the results of Tay et al. on this task, even through correspondence with the authors.

We evaluate SPE (the gated variant) on two efficient Transformer models: the (softmax) Performer (Choromanski et al., 2020), and a Linear Transformer (Katharopoulos et al., 2020) with a ReLU feature map, i.e. choosing $\phi(\cdot) = \max(0, \cdot)$ element-wise in (3).[4] It should be noted that the ReLU feature map does not approximate the softmax kernel, which SPE is designed for (see assumption 8). Nevertheless, it is possible to use SPE with any feature map in practice, allowing us to include Linear Transformer-ReLU as an interesting test of generalization to alternative kernels.

We adopt the configuration of Tay et al., only changing the PE and the batch sizes/learning rates to allow training on limited hardware with similar results. All other hyperparameters are kept identical to the original LRA. It is worth noting that the *Image* models are different from the rest in that they employ a single-layer network and only use the first position for prediction, dramatically limiting their ability to benefit from relative positional information.

Since we observe some variation between different runs, we train and evaluate each model 3 times (except for Performer with convolutional SPE, which is computationally more costly) and report the mean and standard deviation of the results.

The results of the benchmark are given in Table 1. The accuracies achieved by the baseline Linear Transformer-ReLU (`APE`) are similar to or surpass those reported by Tay et al., which is a clear validation of our experimental setup.

Discussion. Results on ListOps are poor overall, with accuracies around 17 %. This complies with Tay et al. (2021), who reasoned that "kernel-based models [e.g. Performer, Linear Transformers] are possibly not as effective on hierarchically structured data," leaving room for improvement. We also hypothesize this is largely due to some known issues with the training data for this task, which unfortunately have not been fixed at the time of this writing.[5]

Regarding performance of SPE, we first notice that the `sineSPE` variant yields the best results on three tasks, which is a strong achievement and validates our approach, especially considering the difficulty of this evaluation benchmark. While it is only marginally better than APE for *ListOps* and *Text*, it is worth mentioning that `sineSPE` combined with the Linear Transformer-ReLU yields an accuracy improvement of $\sim$3 % on *Retrieval* compared to the best result obtained by Tay et al. (2021).

[4] A model named 'Performer' is reported by Tay et al., but communication with the authors revealed it to be in fact equivalent to our Linear Transformer-ReLU, as it does not use random features. To avoid confusion, we refer to this model as such herein.

[5] Currently, the official data loader for ListOps inadvertently strips some characters from the input sequences.

Regarding convSPE, its performance in the LRA is not as remarkable as it is for the music generation experiment reported later in section 3.2. This mitigated result appears somewhat in contradiction with the discussion found in Wang et al. (2021), which presents vanishing attention as a desirable property of PE. On the contrary, we empirically observe that our non-vanishing sinusoidal version sineSPE does behave better in these particular tasks.

Finally, the superior results of APE on *Image* are not unexpected, given the limited ability of these models to exploit relative positions. On the contrary, the relatively good performance of SPE on this task is in fact remarkable, especially considering that the baseline systems for this task use *learnable* APE.

As we will see later in our music generation experiments, there are tasks where our proposed SPE clearly yields remarkable improvements. Here in the LRA, we notice that it does not result in an obvious and systematic boost in performance. This raises interesting considerations:

(i) *The variance of the Monte Carlo estimator might be problematic.* We are enthusiastic about the elegant formulation of stochastic feature maps as in the Performer, which was a strong inspiration. Still, we must acknowledge that their computation relies on a Monte Carlo estimator (15). We suspect that the variance of the estimator might play a role in the final performance in large dimensions, which opens up the direction of exploring variance-reduced estimation methods, rather than plain Monte Carlo.

(ii) *LRA tasks might not benefit from strong (R)PE schemes.* The LRA was designed to compare Transformer *architectures*, filling a gap in this domain and standing as the *de facto* standard, justifying our choice. Still, although PE is known to be important in many cases, it is not known whether it is so in the LRA tasks. We feel that there is room for such a specialized comparison, which is scheduled in our future work, possibly leading to new long-range tasks where PE is critical.

3.2. Pop Piano Music Generation

In our music generation experiments (this subsection and section 3.3), music is represented as sequences of symbols (tokens) and a Performer (Choromanski et al., 2020) is used as an autoregressive language model, which predicts a probability distribution over the next token given the past context. At test time, a new sequence is generated by iteratively sampling the next token, as commonly done in text generation.

Experimental setup. We train Performers for music generation, with 24 layers and 8 heads per layer on a dataset composed of 1 747 pop piano tracks, encoded using the recently proposed *Revamped MIDI-derived format* (REMI; Huang & Yang, 2020). The sequences are composed of *metrical* tokens: bar, subbeat, and tempo, which represent musical timing; and *note* tokens: chord, pitch, duration, and volume, which describe the musical content (see the appendix for more details). We hold out 5% of the songs as the validation set.

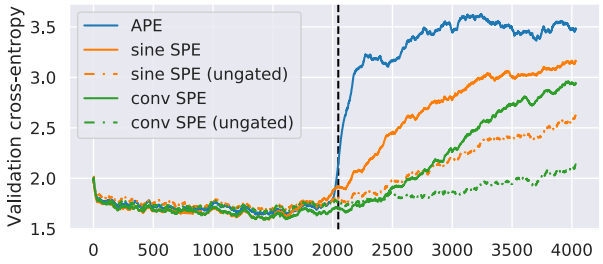


Figure 3. Validation cross-entropy vs. token position on pop piano music generation task. (lower is better; the **black** vertical line indicates the maximum position to which the models are trained.)

We train the models with sequence length $N = 2\,048$, corresponding to $\sim$1 minute of music. The only difference between our models is the PE strategy. We consider baseline APE, as well as SPE: sinusoidal or convolutional, with or without gating, resulting in 5 different models.

Results and discussion. For qualitative assessment, we first display in Figure 1 one attention pattern for each PE model: APE and (gated) sineSPE/convSPE, obtained as an average over 20 from-scratch generations for a chosen (layer, head). More similar plots can be found in appendix. Interestingly, we notice that for early layers, APE attention does not go much beyond training sequence length. This behaviour is not found in SPE variants, which consistently attend to all positions. Another remarkable feature of the proposed model (only displayed in the appendix) is that *gating* as described in section 2.2 visually disables PE altogether for some layers/heads, in which case attention is global.

Since the literature suggests that RPE improves generalization performance (Shaw et al., 2018; Zhou et al., 2019; Rosendahl et al., 2019), we display validation cross-entropy computed with teacher forcing (Williams & Zipser, 1989) in Figure 3, as a function of the target token position. The values would indicate how well the models predict the token at a certain position given the preceding tokens, for tracks in the validation set. We notice that all SPE variants, especially convSPE, behave much better than APE for token positions beyond 2 048. This suggests that SPE inherits this celebrated advantage of RPE (Huang et al., 2018) while being applicable to much longer sequences.

Recently, Wang et al. (2021) defined metrics for the evaluation of PE, suggesting that *translation invariance* and *monotonicity* are desirable properties. The former states that the distances of two arbitrary τ-offset position embeddings should be identical, while the latter states that neighboring

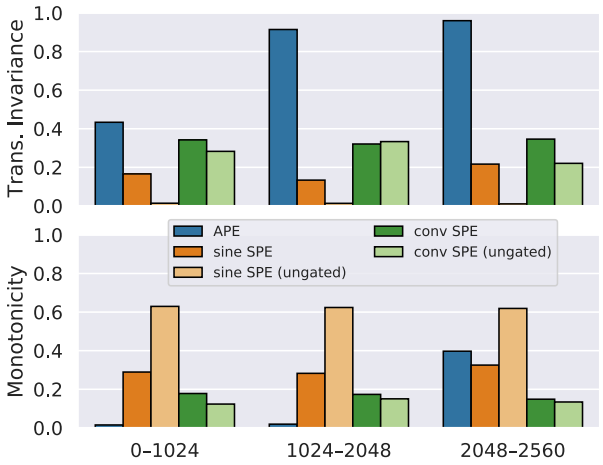


Figure 4. PE evaluation metrics (Wang et al., 2021) for the pop piano music generation task in the 1st layer (lower is better), w.r.t. query positions. Training sequence length is 2 048. Only query-key offsets <128 are considered here. See appendix for details.

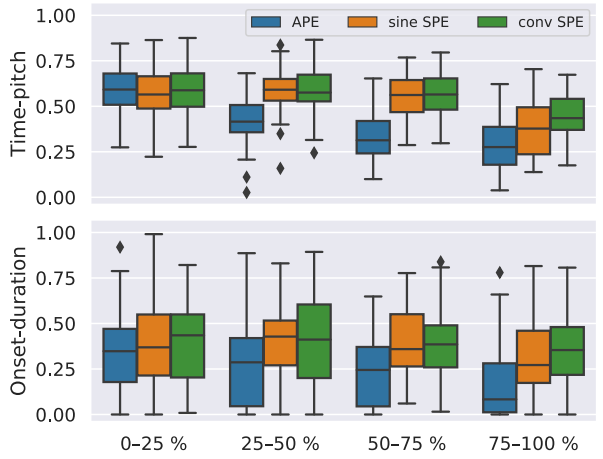


Figure 5. Musical style similarity for groove continuation (higher is better) between output and initial prompt through two musically-motivated metrics, as a function of time in the output. Each data point corresponds to a single musical style.

positions should be assigned with position embeddings that are closer than faraway ones. Following their *identical word probing* methodology, we report these metrics in Figure 4. As expected, SPE variants greatly outperform `APE` in terms of *translation invariance*. However, *monotonicity* does not seem a very relevant criterion in our music application, as can be seen when comparing scores in Figures 3 and 4. It seems that music modeling can benefit from non-vanishing attention patterns. In any case, SPE scores are remarkably stable across positions, contrarily to `APE`, which rapidly degrades beyond the training length.

3.3. Groove Continuation

In this experiment, we evaluate Performers on a *groove continuation* task. After training on a dataset where each example has a uniform style ('groove'), we prime the model with a short *prompt* (2-bar musical fragment) and let it generate a continuation. We then observe whether the generated continuation matches the style of the prompt.

Experimental setup. The models (24-layer Performers with 8 attention heads) are trained on an accompaniment dataset comprising 5 522 samples in 2 761 different musical styles, encoded in a token-based format adopted from Cífka et al. (2020) and detailed in the appendix. All SPE-based models use gating in this experiment. Unlike the previous experiment, which leverages long training sequences, we consider training sequences of length $N = 512$, corresponding to 2–10 bars. At test time, the model is prompted with 2 bars in a style not seen during training and new tokens are sampled to complete the sequence to a length of 1 024, i.e. twice the training length.

We use two musically motivated *style similarity* metrics – *time-pitch* and *onset-duration* proposed by Cífka et al. (2019; 2020) – to quantify the similarity of the generated continuation to the prompt. When listening to the generated music, we perceptually notice a drift in quality along time. For this reason, we divide each generated sample into four successive chunks of identical duration and evaluate them independently. The results are displayed in Figure 5.

Discussion. We clearly see that SPE substantially outperforms `APE` in both metrics. Although `APE` visibly does manage to generate close to the desired style at the beginning of the sequence, this similarity strongly degrades over time. Both `sineSPE` and `convSPE` are much more stable in this regard, confirming the result from section 3.2 that SPE extrapolates better beyond the training sequence length. This matches our informal perceptual evaluation.[6]

This experiment suggests that exploiting a local neighborhood is a robust way to process long sequences. This could appear as contradicting the use of long-range Transformers, but we highlight that *gating* is used here, enabling some heads to exploit long term-attention independently from position. Further comparisons with local attention schemes (e.g. Dai et al., 2019; Hofstätter et al., 2020) could be interesting, although they were not included here due to Tay et al. (2021) suggesting that they are clearly inferior, at least in the LRA setting.

4. Related Work

This paper is concerned with PE (Sukhbaatar et al., 2015), as a way to embed the position of each token as part of its features. This idea is a core ingredient for many subsequent groundbreaking studies (Gehring et al., 2017; Vaswani et al., 2017), and has been the actual topic of many investigations.

[6]Examples: `https://cifkao.github.io/spe/`

Absolute Positional Encoding (APE) based on sinusoids from Vaswani et al. (2017) is the most widely used for Transformer-like architectures. However, PE $\overline{\mathbf{q}}(n)$ and $\overline{\mathbf{k}}(n)$ in (5) can also be trained as in BERT (Devlin et al., 2019; Liu et al., 2019). Although the original Transformer only includes PE at the input layer, it may be included at all layers (Dehghani et al., 2019; Lan et al., 2020).

Relative positional encoding (RPE; Shaw et al., 2018**)** is a way to leverage *relative positions*. It came with a $\mathcal{O}(N^2D)$ space complexity, which was reduced to $\mathcal{O}(N^2)$ in Huang et al. (2018); He et al. (2020). Considering log-distances was proposed in Raffel et al. (2020). Several variants for RPE were introduced (Huang et al., 2020; Wang et al., 2021). They all apply learned RPE in the attention domain. Using fixed embedding functions was also considered for RPE (Pham et al., 2020), and masking RPE is used in Kim et al. (2020) to promote local attention.

Keys-domain vs attention domain. Doing PE in the keys domain introduces position-content cross terms that are advocated as noisy and not beneficial in Ke et al. (2020) and replaced by *Untied* attention, i.e. PE in the attention domain. This is also called *disantangled attention* in He et al. (2020) and already proposed in Tsai et al. (2019) through *separable* content-position attention kernels. All of these studies require the explicit computation and storage of $\mathbf{A}$.

Non-integer positions were considered for structured inputs. Tree-based PE was proposed both for APE (Shiv & Quirk, 2019; Xiao et al., 2019; Ma et al., 2019) and RPE (Omote et al., 2019). Positional encoding of robots within arbitrary polygons is found in Bose et al. (2019).

Dynamical models for PE. Attention for machine translation was introduced in Bahdanau et al. (2016), which was retrospectively understood in Ke et al. (2020) as using recurrent neural nets (RNN) for PE. In Chen et al. (2018), the hidden states of encoder RNNs are said to contain enough position information to skip explicit PE. Neishi & Yoshinaga (2019) builds on this view, but explicitly describes the idea for the first time. Their contribution is to replace the additive PE in (5) by an RNN. In the same vein, Liu et al. (2020) generates PE using (neural) ordinary differential equations.

Convolutional contexts. Our `convSPE` variant involves convolving random noise. First, this can be related to Mohamed et al. (2019), who use convolutional neural networks for queries and keys computation. Second, the connections between convolutions and stationary processes have recently been highlighted by Xu et al. (2020) as enforcing PE.

Multiplicative PE. Various levels of content-position interactions are formalized in (Tsai et al., 2019). Multiplicative strategies were proposed for both RPE (Huang et al., 2020) and APE (Dai et al., 2019). The latter was generalized in Tsai et al. (2019). All these require the explicit computation of the attention matrix. Wang et al. (2020a) presents a scheme that is close to our sinusoidal variant, but without the stochastic part that is the key to go from (14) to (15).

The limits of APE and RPE were highlighted by some authors. In Wang & Chen (2020), the best performing models exploit both absolute *and* relative positions. In Irie et al. (2019) and Tsai et al. (2019), it is found that removing APE altogether in the causal decoder part of Transformer-based architectures leads to comparable/better performance. It is also not clear which one is best between incorporating PE in the raw input signal (and hence propagating it through the *value* entries) or using it anew on the queries and keys only, as we do. Our choice is backed by Tsai et al. (2019).

5. Conclusion

We propose a new Stochastic Positional Encoding (SPE), based on filtering random noise. As we show, the procedure generalizes relative PE and is a principled means to enforce any prescribed (but trained) cross-covariance structure, which we demonstrated should be the central concern in dot-product attention. In our experiments, we show that SPE brings an interesting gain in performance for large-scale transformer models (Choromanski et al., 2020; Katharopoulos et al., 2020), as compared to classical (sinusoidal) PE. This was expected, because RPE (Shaw et al., 2018) is often advocated as beneficial. However, no way to incorporate it for long sequences was available so far and this is the core contribution of this paper. The natural future directions for our study are (i) *Signal-dependent PE* that incorporates the input sequence as an additional input for SPE, (ii) *Nonstationary PE* that utilizes both relative and absolute positions, (iii) Extending our approach to *arbitrary attention kernels*, e.g. defined implicitly through their (random) mappings as in (4). Indeed, SPE as it is presented here holds theoretically for dot-product attention kernels only, but our results given in Table 1 suggest that this generalizes, asking an interesting research question.

Acknowledgements

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers) and in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

We would like to thank Yi Tay, Mostafa Dehghani and Philip Pham for their help with troubleshooting the Long-Range Arena, and Krzysztof Choromanski for clarifications about the Performer.

References

AlQuraishi, M. AlphaFold at CASP13. *Bioinformatics*, 35 (22):4862–4865, 2019.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473 [cs, stat]*, May 2016. URL http://arxiv.org/abs/1409.0473. arXiv: 1409.0473.

Bello, I. LambdaNetworks: Modeling long-range interactions without attention. In *Proc. Int. Conf. Learning Representations*, 2020.

Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., and Widmer, G. Madmom: A new Python audio and music signal processing library. In *Proc. ACM International Multimedia Conf.*, pp. 1174–1178, 2016.

Bose, K., Adhikary, R., Kundu, M. K., and Sau, B. Positional encoding by robots with non-rigid movements. *arXiv:1905.09786 [cs]*, May 2019. URL http://arxiv.org/abs/1905.09786. arXiv: 1905.09786.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Parmar, N., Schuster, M., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. *arXiv:1804.09849 [cs]*, April 2018. URL http://arxiv.org/abs/1804.09849. arXiv: 1804.09849.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse Transformers. *arXiv:1904.10509 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1904.10509. arXiv: 1904.10509.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with Performers. *arXiv:2009.14794 [cs, stat]*, September 2020. URL http://arxiv.org/abs/2009.14794. arXiv: 2009.14794.

Cífka, O., Şimşekli, U., and Richard, G. Supervised symbolic music style translation using synthetic data. In *Proc. International Society for Music Information Retrieval Conf.*, pp. 588–595, 2019. doi: 10.5281/zenodo.3527878. URL https://doi.org/10.5281/zenodo.3527878.

Cífka, O., Şimşekli, U., and Richard, G. Groove2Groove: One-shot music style transfer with supervision from synthetic data. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 28:2638–2650, 2020. doi: 10.1109/TASLP.2020.3019642. URL https://hal.archives-ouvertes.fr/hal-02923548.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal Transformers. *arXiv:1807.03819 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1807.03819. arXiv: 1807.03819.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

Donahue, C., Mao, H. H., Li, Y. E., Cottrell, G. W., and McAuley, J. Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. In *ISMIR*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Engel, J., Hantrakul, L., Gu, C., and Roberts, A. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. *arXiv:1705.03122 [cs]*, July 2017. URL http://arxiv.org/abs/1705.03122. arXiv: 1705.03122.

Genton, M. G. and Kleiber, W. Cross-covariance functions for multivariate geostatistics. *Statistical Science*, pp. 147–163, 2015.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., and Eck, D. Onsets and Frames: Dual-objective piano transcription. In *Proc. Int. Society for Music Information Retrieval Conf.*, 2018.

He, P., Liu, X., Gao, J., and Chen, W. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv:2006.03654 [cs]*, June 2020. URL http://arxiv.org/abs/2006.03654. arXiv: 2006.03654.

Hofstätter, S., Zamani, H., Mitra, B., Craswell, N., and Hanbury, A. Local self-attention over long text for efficient document retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2021–2024, 2020.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *Proc. International Conference on Learning Representations*, 2019.

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., and Yang, Y.-H. Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proc. AAAI Conf. Artificial Intelligence*, 2021.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music Transformer. *arXiv:1809.04281 [cs, eess, stat]*, December 2018. URL http://arxiv.org/abs/1809.04281. arXiv: 1809.04281.

Huang, Y.-S. and Yang, Y.-H. Pop Music Transformer: Generating music with rhythm and harmony. In *Proc. ACM International Multimedia Conf.*, 2020.

Huang, Z., Liang, D., Xu, P., and Xiang, B. Improve Transformer models with better relative position embeddings. *arXiv preprint arXiv:2009.13658*, 2020.

Irie, K., Zeyer, A., Schlüter, R., and Ney, H. Language modeling with deep Transformers. *Proc. Interspeech*, pp. 3905–3909, 2019. doi: 10.21437/Interspeech.2019-2225. URL http://arxiv.org/abs/1905.04226. arXiv: 1905.04226.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast autoregressive Transformers with linear attention. In *Proc. Int. Conf. Machine Learning*, pp. 5156–5165, 2020.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. *arXiv:2006.15595 [cs]*, July 2020. URL http://arxiv.org/abs/2006.15595.

Kim, J., El-Khamy, M., and Lee, J. T-GSA: Transformer with Gaussian-weighted self-attention for speech enhancement. *arXiv:1910.06762 [cs, eess]*, February 2020. URL http://arxiv.org/abs/1910.06762. arXiv: 1910.06762.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kucerovsky, D., Mousavand, K., and Sarraf, A. On some properties of toeplitz matrices. *Cogent Mathematics*, 3(1), 2016. doi: 10.1080/23311835.2016.1154705. URL http://doi.org/10.1080/23311835.2016.1154705.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv:1909.11942 [cs]*, February 2020. URL http://arxiv.org/abs/1909.11942. arXiv: 1909.11942.

Liu, X., Yu, H.-F., Dhillon, I., and Hsieh, C.-J. Learning to encode position for Transformer with continuous dynamical model. *arXiv:2003.09229 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.09229. arXiv: 2003.09229.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692 [cs]*, July 2019. URL http://arxiv.org/abs/1907.11692. arXiv: 1907.11692.

Ma, C., Tamura, A., Utiyama, M., Sumita, E., and Zhao, T. Improving neural machine translation with neural syntactic distance. In *Proc. Conf. North American Chapter of the Association for Computational Linguistics*, pp. 2032–2037, 2019. doi: 10.18653/v1/N19-1205. URL http://aclweb.org/anthology/N19-1205.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proc. Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011. URL https://www.aclweb.org/anthology/P11-1015.

Matheron, G. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

Mohamed, A., Okhonko, D., and Zettlemoyer, L. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019.

Nadaraya, E. A. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Nangia, N. and Bowman, S. ListOps: A diagnostic dataset for latent tree learning. In *Proc. Conf. North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99, 2018. doi: 10.18653/v1/N18-4013. URL https://www.aclweb.org/anthology/N18-4013.

Neishi, M. and Yoshinaga, N. On the relation between position information and sentence length in neural machine translation. In *Proc. Conf. Computational Natural Language Learning*, pp. 328–338, 2019. doi: 10.18653/v1/K19-1031. URL https://www.aclweb.org/anthology/K19-1031.

Omote, Y., Tamura, A., and Ninomiya, T. Dependency-based relative positional encoding for Transformer NMT. In *Proc. Natural Language Processing in a Deep Learning World*, pp. 854–861, 2019. ISBN 978-954-452-056-4. doi: 10.26615/978-954-452-056-4_099. URL https://acl-bg.org/proceedings/2019/RANLP2019/pdf/RANLP099.pdf.

Pham, N.-Q., Ha, T.-L., Nguyen, T.-N., Nguyen, T.-S., Salesky, E., Stueker, S., Niehues, J., and Waibel, A. Relative positional encoding for speech recognition and direct translation. *arXiv:2005.09940 [cs, eess]*, May 2020. URL http://arxiv.org/abs/2005.09940.

Radev, D. R., Muthukrishnan, P., Qazvinian, V., and Abu-Jbara, A. The ACL anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, January 2013. doi: 10.1007/s10579-012-9211-2. URL https://doi.org/10.1007/s10579-012-9211-2.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text Transformer. *arXiv:1910.10683 [cs, stat]*, July 2020. URL http://arxiv.org/abs/1910.10683. arXiv: 1910.10683.

Rosendahl, J., Tran, V. A. K., Wang, W., and Ney, H. Analysis of positional encodings for neural machine translation. In *Proc. IWSLT*, 2019.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing Transformers. *arXiv:2003.05997 [cs, eess, stat]*, October 2020. URL http://arxiv.org/abs/2003.05997. arXiv: 2003.05997.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *arXiv:1803.02155 [cs]*, April 2018. URL http://arxiv.org/abs/1803.02155. arXiv: 1803.02155.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. Efficient attention: Attention with linear complexities. *arXiv:1812.01243 [cs]*, November 2020. URL http://arxiv.org/abs/1812.01243. arXiv: 1812.01243.

Shiv, V. and Quirk, C. Novel positional encodings to enable tree-based Transformers. In *Proc. Advances in neural information processing systems*, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. End-to-end memory networks. *arXiv:1503.08895 [cs]*, November 2015. URL http://arxiv.org/abs/1503.08895. arXiv: 1503.08895.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long Range Arena: A benchmark for efficient Transformers. In *Proc. Int. Conf. Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. Transformer dissection: An unified understanding for Transformer's attention via the lens of kernel. In *Proc. Conf. Empirical Methods in Natural Language Processing and Int. Joint Conf. Natural Language Processing*, pp. 4335–4344, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proc. Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vetterli, M., Kovačević, J., and Goyal, V. K. *Foundations of signal processing*. Cambridge University Press, 2014.

Vořechovský, M. Simulation of simply cross correlated random fields by series expansion methods. *Structural safety*, 30(4):337–363, 2008.

Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., and Simonsen, J. G. Encoding word order in complex embeddings. *arXiv:1912.12333 [cs]*, June 2020a. URL http://arxiv.org/abs/1912.12333. arXiv: 1912.12333.

Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q., and Simonsen, J. G. On position embeddings in BERT. In *Proc. Int. Conf. Learning Representations*, 2021. URL https://openreview.net/forum?id=onxoVA9FxMw.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv:2006.04768 [cs, stat]*, June 2020b. URL http://arxiv.org/abs/2006.04768. arXiv: 2006.04768.

Wang, Y.-A. and Chen, Y.-N. What do position embeddings learn? An empirical study of pre-trained language model positional encoding. In *Proc. Conf. Empirical Methods in Natural Language Processing*, 2020.

Watson, G. S. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Wu, Q., Lan, Z., Gu, J., and Yu, Z. Memformer: The memory-augmented Transformer. *arXiv:2010.06891 [cs]*, October 2020. URL http://arxiv.org/abs/2010.06891. arXiv: 2010.06891.

Xiao, F., Li, J., Zhao, H., Wang, R., and Chen, K. Lattice-Based Transformer encoder for neural machine translation. *arXiv:1906.01282 [cs]*, June 2019. URL http://arxiv.org/abs/1906.01282. arXiv: 1906.01282.

Xu, R., Wang, X., Chen, K., Zhou, B., and Loy, C. C. Positional encoding as spatial inductive bias in gans. *arXiv preprint arXiv:2012.05217*, 2020.

Xue, H. and Salim, F. D. Trailer: Transformer-based time-wise long term relation modeling for citywide traffic flow prediction. *arXiv preprint arXiv:2011.05554*, 2020.

Yang, Z., Xie, L., and Stoica, P. Vandermonde decomposition of multilevel Toeplitz matrices with application to multidimensional super-resolution. *IEEE Transactions on Information Theory*, 62(6):3685–3701, 2016.

Zhou, P., Fan, R., Chen, W., and Jia, J. Improving generalization of transformer for speech recognition with parallel schedule sampling and relative positional embedding. *arXiv preprint arXiv:1911.00203*, 2019.

Relative Positional Encoding for Transformers with Linear Complexity
Supplementary Material

Introduction

This document comprises additional information that could not be included in the paper due to space constraints. It is structured as follows. In appendix A, we provide some further theoretical developments. In appendix B, we detail the experimental setup on the Long Range Arena. In appendix C, we detail our music generation experiments. Finally, we provide additional results in appendix D.

Our source code is available at:

https://github.com/aliutkus/spe/

See also the companion website:

https://cifkao.github.io/spe/

A. Theory

A.1. Convolutional SPE Leads to Vanishing Attention

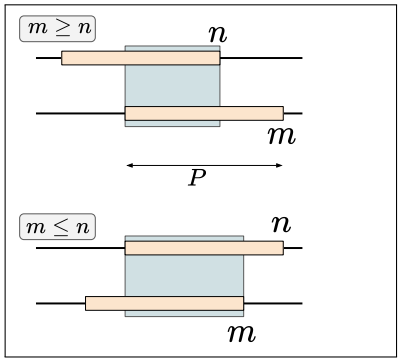


Figure 6. If $\mathbf{\Phi}_d^Q$ and $\mathbf{\Phi}_d^K$ have length P, $\overline{Q}_d$ and $\overline{K}_d$ for convolutional SPE depend on the noise Z_d over the intervals $[m-P:m]$ and $[n-P:n]$, respectively. Their correlation depends only on the shaded area, due to whiteness of Z_d. Whenever $|m-n|>P$, the two signals are uncorrelated.

In the main document, we claim that the convolutional variant leads to vanishing attention. We shortly prove this claim here. For ease of notation, the proof is given in the 1D case, but extends trivially to higher dimensions. The core idea is illustrated in Figure 6. Convolutional SPE yields:

$$\overline{Q}_d(m,r) = \sum_{p=0}^{P} Z_d(m-p,r)\phi_d^Q(p),$$
$$\overline{K}_d(n,r) = \sum_{p=0}^{P} Z_d(n-p,r)\phi_d^K(p),$$

where Z_d is a white Gaussian noise process, i.e. $\mathbb{E}[Z_d(m,r)Z_d(m',r)] = \delta_{mm'}$. Omitting the dependency on r for notational convenience (all realizations are independent), we can compute the positional attention as:

$$\begin{aligned}
\mathcal{P}_d(m,n) &= \mathbb{E}\left[\overline{Q}_d(m)\overline{K}_d(n)\right] \\
&= \mathbb{E}\left[\sum_{p,\tau=0}^{P} z_d(m-\tau)z_d(n-p)\phi_d^Q(\tau)\phi_d^K(p)\right] \\
&= \mathbb{E}\left[\sum_{p=0}^{P} z_d(n-p)^2\phi_d^Q(p+m-n)\phi_d^K(p)\right] \\
&= \sum_{p=0}^{P} \phi_d^Q(p+m-n)\phi_d^K(p),
\end{aligned}$$

where only the (p,τ) values such that $n-p=m-\tau$ remain, all other cross terms $\mathbb{E}[Z_d(m)Z_d(m'\neq m)]$ disappearing due to whiteness of Z_d. Filters are taken as 0-valued outside of $[0:P]$. As can be seen, whenever $|m-n|>P$, we get $\mathcal{P}_d(m,n)=0$, because $\phi_d^K(p+(m-n))=0$. □

A.2. Complexity

In this section, we detail the additional complexity caused by the proposed SPE method.

- **Sinusoidal SPE** first requires the computation of the modulation matrices $\mathbf{\Omega}$ for each feature dimension $d = 1\dots D$, which has a $\mathcal{O}(2NK)$ complexity. Then, this matrix must be multiplied by the noise matrix $\mathbf{Z}_d$ with shape $2K \times R$, leading to an overall complexity of $\boldsymbol{\mathcal{O}(DRNK^2)}$. Since K is typically very small in our experiments, `SineSPE` can be seen as quite light in terms of both time and space complexity.
- **Convolutional SPE** involves drawing a new noise signal $\mathbf{z}_{d,:,r}$ of length N for each d and r, and convolving it with the filters $\boldsymbol{\phi}_d^Q$ and $\boldsymbol{\phi}_d^K$, whose length is written P.
  In the 1D case, this leads to an overall time complexity of $\boldsymbol{\mathcal{O}(DRNP)}$, which can be replaced by $\mathcal{O}(DRN\log N)$

when operating the convolutions in the frequency domain, which is advantageous for long filters.
In higher dimensions, say 2D, this becomes $\mathcal{O}(DRN_1N_2P_1P_2)$ in the original domain and $\mathcal{O}(DRN_1N_2 \log N_1 \log N_2)$ in the frequency domain, where (N_1, N_2) and (P_1, P_2) are the shapes of noise and filters, respectively.
- The bottleneck of **gating** is the generation of random noise $\boldsymbol{\epsilon}_d$, which has complexity $\mathcal{O}(DR)$.

Note that this complexities of course must be multiplied by the number of heads considered, up to 8 in our experiments.

As can be seen, the complexities of the sinusoidal and convolutional variants are similar, depending on the length P of the filters and the number K of sinusoids.
Still, other aspects come into the play. First, the convolutional version requires generating noise whose size scales as N, while the sinusoidal version requires much smaller $2K$-large noise matrices. Second, only a very small number of sinusoids was required in our experiments, whereas the convolutional version required longer contexts, so that we often had $2K \ll P$ in practice. Finally, although this may change in the near future, deep learning frameworks like PyTorch do not easily integrate convolutions in the frequency domain.

Sample-wise noise sharing. In practice, SPEs do not need to be drawn anew for each example. The most straightforward trick to reduce memory and computational footprint of the method is to share $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ among all examples in each mini-batch, as we do in all our experiments. This can bring significant memory savings when SPE is used as a drop-in addition to networks trained with large batch sizes.

B. Experimental Setup: Long-Range Arena

An overview of the Long-Range Arena (Tay et al., 2021) tasks is given in table 2. We do not include *Pathfinder* (a synthetic image classification task) or its harder variant *Pathfinder-X* in this paper as we were unable to reproduce the results of Tay et al. on this task. All the datasets are described in detail in Tay et al. and available from the official LRA repository.[7]

In all LRA experiments, we employ gated SPE with $R \in \{32, 64\}$. We consistently use $K = 10$ for sinusoidal (periodic) SPE and filters of length 128 for convolutional SPE. For convolutional SPE, we share $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ across all layers (but not across attention heads); for sinusoidal SPE, $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ are unique to each layer and head; in both cases, layer-specific gating is employed. Baseline experiments employ the same absolute positional encodings as Tay et al. (learnable APE for Image and sinusoidal APE for the remaining tasks). In models employing SPE, APE is removed.

The numbers of parameters of the models presented in the main document are shown in Table 3. We can see that SPE-based models have at most 3.1 % more parameters than the baselines. In the Image column, the numbers for SPE-based models are about 50 % lower due to the fact that the baselines on this task employ learnable APE.

We use code from the official LRA repository, including the authors' Transformer implementation, modified as necessary to incorporate SPE. We keep the same training configuration as provided by the LRA authors, but decrease the batch sizes (from 256 to 96 for Image and from 32 to 8 for the rest) and learning rates so as to fit within 16 GB of GPU memory. Our modified code and configuration files are available in our source code repository.

B.1. Resource usage

The typical training times of the LRA models are displayed in Table 4. Note that the times may not be comparable across models or tasks due to evaluation (which may be time-consuming) being done more frequently in some runs than others.

The total training time was 1 405 h (189 runs in total), out of which 273 h (61 runs) were spent on attempts to reproduce the results of Tay et al. (2021) using Performer-softmax, Linear Transformer-ReLU and vanilla Transformer. Some of these preliminary experiments were distributed over 1–3 Tesla V100 GPUs with 32 GB of memory each. The final models were all trained on a single Tesla V100 or P100 GPU with 16 GB of memory.

C. Experimental Setup: Music Generation

Our music Performers are implemented using the `pytorch-fast-transformers` package,[8] modified as necessary to incorporate SPE. The modified code and configuration files are available in our code repository.

All models have 24 layers with model dimension 512, 8 attention heads and 2 048 feed-forward units, which amount to ∼80 million trainable parameters. In models that use SPE, $\overline{\mathbf{Q}}$ and $\overline{\mathbf{K}}$ are shared across all layers (but not across attention heads); layer-specific gating is employed for models trained with gated SPE.

The models are trained with the Adam optimizer. We schedule the learning rate with linear warmup, followed by cosine decay. Full details of hyperparameters can be found in the provided configuration files.

[7] https://github.com/google-research/long-range-arena

[8] https://github.com/idiap/fast-transformers

Table 2. Long-Range Arena classification tasks used in this paper.

Name	Dataset	Input	Length	Goal	# classes
ListOps	ListOps	expression with operations on lists of digits	2 k	evaluate expression	10
Text	IMDB	movie review as byte string	8 k	classify sentiment	2
Retrieval	AAN	pair of articles as byte strings	2×4 k	detect citation link	2
Image	CIFAR10	8-bit gray-scale 32×32 image as byte string	1 k	recognize object	10

Table 3. Numbers of parameters of LRA models, identical for both Performer-softmax and Linear Transformer-ReLU.

	ListOps	Text	Retrieval	Image
Baseline (APE)	19 982 858	3 486 722	1 087 618	248 458
+ `sineSPE`	20 078 090	3 518 466	1 103 490	119 242
+ `convSPE`	20 117 002	3 553 282	1 120 898	133 706

C.1. Pop Piano Music Generation

Training data. The pop piano MIDI dataset we use is derived from the one provided in Hsiao et al. (2021), open-sourced on GitHub.[9] It consists of 1,747 pure piano performances of various Japanese, Korean, and Western pop songs, amounting to a total duration of ∼100 hours. All the songs are in 4/4 time signature, namely four beats per bar (measure). We leave 5% (87 songs) as the validation set.

According to Hsiao et al. (2021), the piano performances are originally collected from the Internet in the MP3 (audio) format. Hsiao et al. further employed *Onsets and Frames* piano transcription (Hawthorne et al., 2018), `madmom` beat tracking tool (Böck et al., 2016), and `chorder` rule-based chord detection[10] to transcribe the audio into MIDI format with tempo, beat, and chord information.

Data representation. The representation adopted here is largely identical to the *Revamped MIDI-derived* (REMI) encoding by Huang & Yang (2020), except that an extended set of `chord` tokens (described below) is used. REMI encodes a piano piece into a sequence composed of two types, *metrical* and *note*, of tokens. The *metrical* tokens are:

- `bar`: Marks the start of a musical bar.
- `subbeat`: Marks the musical timing within a bar. A bar is divided into 16 `subbeats`, which is equivalent to 4 beats. This symbolic timing provides an explicit time grid for sequence models to model music.
- `tempo`: Determines the pace (in beats per minute, or *bpm*) at which the piece is played, varied per bar. The range of `tempo` tokens is $[32, 224]$ bpm, in steps of 3 bpm for quantization.

The *note* tokens are:

- `pitch`: Marks a note played. The 88 `pitch`-es correspond to each key on the piano.
- `duration`: Denotes the length of a played note, ranging from $1/2$ to 16 subbeats, in steps of $1/2$ subbeat.
- `volume` (or, velocity): Denotes how loud a note is played. A total of 24 `volume` levels are considered.
- `chord`: Marks a change on the accompanying chord. Each `chord` is described by its *root note* and *quality*, e.g., `C-Maj7`, `E-min`. A total of 133 distinct `chord` tokens are found in the dataset.

Please note that a single note played is represented by a co-occurring triple of (`pitch`, `duration`, `volume`). The aforementioned tokens constitute a vocabulary of size ∼340 for our REMI encoding. On average, we need a sequence with 5 300 tokens to represent a song.

Training and inference. In each training epoch, we randomly crop a segment of length 2 048 from each sample, and shift the pitches of the entire segment by -6 to 6 semitones randomly (this is called *transposition* in music) as data augmentation. We use batch size $= 4$, and set the learning rate to 0.0001 for APE and 0.0002 for all SPE models. For `sineSPE`, we choose the number of sines $K = 5$; for `convSPE`, the convolutional filter size is set to be $128, 512$ for the gated and ungated variants respectively.

Detailed resource usage of each model is shown in Table 5.

During inference, we employ *nucleus sampling* (Holtzman et al., 2019) with $p = 0.9$ and softmax temperature $t = 1.2$. No post-processing on enforcing the grammatical correctness of the generated sequence is done.

Validation loss of the models trained on this task is listed in Table 6. On this metric, our `convSPE` variant performs the

[9] https://github.com/YatingMusic/compound-word-transformer

[10] https://github.com/joshuachang2311/chorder

Table 4. Training times for LRA models (hours). Numbers in parentheses are from Tesla P100 GPUs, the rest from Tesla V100 GPUs.

	ListOps	Text	Retrieval	Image
Performer-softmax	1.1	4.8	1.2	4.8
Performer-softmax + `sineSPE`	(4.2)	11.7	2.9	5.0
Performer-softmax + `convSPE`	8.9	23.2	21.9	5.3
Linear Transformer-ReLU	0.6	(3.2)	0.7	4.8
Linear Transformer-ReLU + `sineSPE`	2.0	6.8	2.1	5.0
Linear Transformer-ReLU + `convSPE`	15.0	18.6	19.0	5.3

Table 5. Resource usage of models trained on pop piano music generation, on a Tesla V100 GPU with 32GB of memory. # of epochs and time to the checkpoint with the lowest validation loss are displayed. (`ug`: trained without SPE gating.)

	# epochs	Time	Memory
`APE`	72	9.74 h	14.34 GB
`sineSPE`	78	17.92 h	29.80 GB
`sineSPE (ug)`	78	16.31 h	18.29 GB
`convSPE`	80	28.02 h	30.01 GB
`convSPE (ug)`	68	24.76 h	18.99 GB

Table 6. Validation cross-entropy for models trained for pop piano music generation (mean and standard deviation) over all sequences. (`ug`: trained without SPE gating). *Trained*: pos $\leq$ 2 048, *Extrapolation*: 2 048 $<$ pos $\leq$ 3 072.

Positions	*Trained*	*Extrapolation*
`APE`	1.721 $\pm$ 0.148	3.215 $\pm$ 0.200
`sineSPE`	1.694 $\pm$ 0.148	2.396 $\pm$ 0.359
`sineSPE (ug)`	1.754 $\pm$ 0.146	1.965 $\pm$ 0.170
`convSPE`	**1.685** $\pm$ 0.151	1.932 $\pm$ 0.225
`convSPE (ug)`	1.733 $\pm$ 0.145	**1.805** $\pm$ 0.163

best both within the trained positions and on extrapolation.

C.2. Groove Continuation

Training data. The Groove2Groove MIDI dataset[11] consists of accompaniments generated by the Band-in-a-Box software (BIAB).[12] We only use the training section of the Groove2Groove MIDI dataset and perform a custom training/validation/test split such that each section contains a unique set of BIAB styles (2 761 for training and 50 each for validation and testing). The code necessary to download, pre-process and split the dataset is included in the repository.

We convert each accompaniment to a trio consisting of bass, drums and another randomly selected accompaniment track (e.g. piano, guitar). We then perform random data augmentation by skipping measures at the beginning, dropping some of the instruments, and transposition (pitch-shifting by -5 to $+5$ semitones). All randomization is done anew in each epoch.

Data representation. We use a representation similar to the one proposed by Cífka et al. (2020), but adapted to a multi-track (multi-instrument) setting. Specifically, we encode a piece of music as a sequence of the following types of event tokens, each with two integer arguments:

- `note_on(track, pitch)`: Begins a new note at the given pitch (0–127).
- `note_off(track, pitch)`: Ends the note at the given pitch (0–127).
- `time_shift(beats, offset)`: Advances current time by a given number of beats and then sets the offset within the beat, given as the number of ticks from its beginning (0–11). Maximum possible shift is `(2, 0)`.

The track numbers range from 1 to 3, where 1 is always bass and 2 is always drums. The vocabulary of the model then consists of 794 tokens (3×128 note-ons, 3×128 note-offs, 24 time shifts, and 2 beginning-/end-of-sequence markers).

The main differences to the representation described in Section C.1 are a more compact encoding of timing, no representation of musical dynamics (for simplicity), and support for multiple tracks (not originally proposed by Cífka et al., 2020 but introduced here inspired by Donahue et al., 2019).

Training and inference. During training, each example is pre-processed and encoded as described above and the resulting token sequence is truncated to a length of 512. We train each model for a total of 24 epochs.

At test time, we sample with a softmax temperature of 0.6. We disallow sampling tokens that would result in invalid sequences (i.e. spurious note-offs, backward time shifts) in order to ensure that the generated sequence can be correctly decoded.

[11] http://doi.org/10.5281/zenodo.3958000
[12] https://www.pgmusic.com/

Various training details. Hyperparameter tuning was mostly performed in preliminary experiments (∼100 runs); these were mostly done on other variants of the dataset and with different sequence lengths (ranging from 256 to 20 k); this includes experiments discarded due to bugs discovered during or after training. Learning rates between 0.0001 and 0.0008 and batch sizes between 1 and 24 were considered. For SPE, we considered both the gated and ungated variants with as many realizations as fit in memory (between 16 and 64). Model selection was based on validation loss and informal perceptual evaluation. Only a minimal attempt at further learning rate tuning was made for the final set of models with length 512, which did not appear to be particularly sensitive to it, and we chose to keep the initial learning rate 0.0004, which was found to perform well in all cases.

The models included in the main document – `APE`, `sineSPE` and `convSPE` – all use a batch size of 10 and finished training in about 3 h, 5 h and 6 h, respectively, using 9.7 GB, 14.4 GB and 14.8 GB of GPU memory. The total training time including all preliminary experiments was 852 hours.

Evaluation metrics. We use the objective metrics proposed by Cífka et al. (2019; 2020) to measure the style similarity between the generated continuation and the file from which the prompt was extracted. Given two pieces of music, each metric gathers musical event statistics of the two pieces in histograms called *style profiles*, and then computes the cosine similarity between them.

The two metrics used here, *onset-duration* and *time-pitch*, differ in what kind of events they use to construct the style profile:

- The **onset-duration** profile is defined as a 2D histogram relating note onset positions to note durations. More precisely, for all notes a in a piece of music, it records a tuple of the form

  $$(\text{start}(a) \bmod 4,\ \text{end}(a) - \text{start}(a)) \in [0, 4) \times [0, 2),$$

  where $\text{start}(a)$ and $\text{end}(a)$ refer to the onset and offset time of a in beats. The expression $\text{start}(a) \bmod 4$ then represents the position of the note onset relative to the current bar, since all examples in the dataset are in a 4-beat meter. These tuples are gathered in 24×12 histogram bins (24 for onset time and 12 for duration).
- The **time-pitch** profile is also obtained as a 2D histogram, this time capturing time differences and pitch differences (intervals) between notes. The tuples it considers have the form

  $$(\text{start}(b) - \text{start}(a),\ \text{pitch}(b) - \text{pitch}(a)) \in [0, 4) \times \{-20, -19, \ldots, 20\},\ a \neq b,$$

  where a, b is a pair of notes and $\text{pitch}(\cdot)$ represents the pitch of a note as its MIDI note number (the number of semitones from C_{-1}). The histogram has 24×41 bins (24 for time lags between 0 and 4 beats and 41 bins for intervals between -20 and 20 semitones).

In both cases, the 2D histograms are flattened to vectors before computing cosine similarities.

D. Additional Results

D.1. Attention Visualization: Music Generation

In this section, we display attention patterns produced by our pop piano music generation models.

Learned positional templates. We share the SPE modules across all layers of the Performer, but not across the attention heads, resulting in 512 learned positional kernels $\mathcal{P}_d$) (*number of heads* $\times$ *key dimensions per head*. In Figure 7, we display 16 randomly picked resulting templates $\mathbf{P}_d$ for both `sineSPE` and `convSPE`, trained with gating. Details of the two variants are:

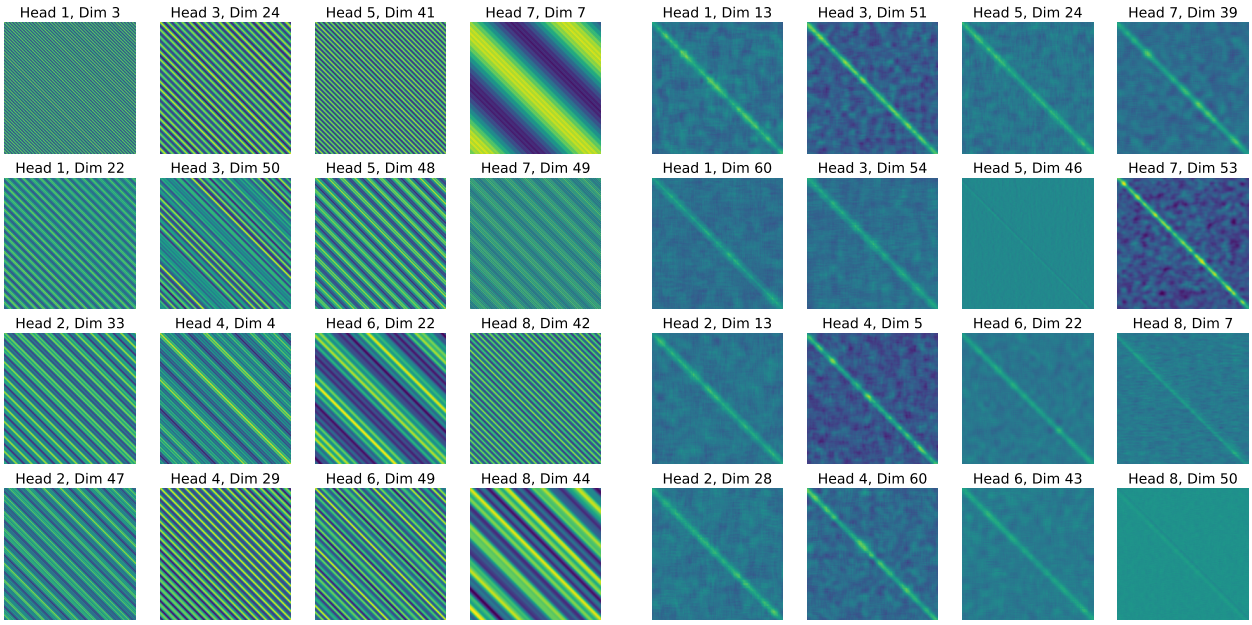
- `sineSPE`: We set the number of sines $K = 5$.
- `convSPE`: We use filters of size 128.

In accordance with the definition, all of the visualizations are plotted with the equation $\mathbf{P}_d = \overline{\mathbf{Q}}_d \overline{\mathbf{K}}_d^\top$, which we never need to explicitly compute for linear transformers. From Figure 7, we can observe that `sineSPE` learns to exploit a wide range of frequencies, and that `convSPE` is effective within small query-key offsets corresponding to the filter size, as expected.

Full Attention. Although the full attention matrix $\mathbf{A}$ is not computed in linear transformers, we can still obtain it *offline* by multiplying queries and keys through either $\mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{D})$ (in the case of APE, where D is the key dimensions per head), or $\mathbf{A} = \exp(\widehat{\mathbf{Q}}\widehat{\mathbf{K}}^\top/\sqrt{R})$ (in the case of SPEs); then apply row-wise softmax operation on $\mathbf{A}$ as normalization.

Here, we present the (softmax-ed) attention matrices in the 1st, 3rd, 12th, 20th, and 24th (last) layers of all the five models trained on pop piano music generation in Figures 8–12. These are computed from one of each model's random from-scratch music generations. To examine the models' extrapolation ability, we let them generate a sequence of length $3\,072$, while the training sequence length is only $2\,048$. The attention matrices are lower-triangular due to causal masking. For better visualization, the color of each pixel is adjusted through $\min\{1, {a_{mn}}^{0.4}/0.02^{0.4}\}$ in the plots, where $a_{mn} \in [0, 1]$ is the softmax-ed attention score.

Figure 8 reveals a major drawback of `APE`: the attention of



(a) `sineSPE` (b) `convSPE`

Figure 7. Examples of $\mathbf{P}_d$ learned by SPE. X- and Y-axes are *key* and *query* positions respectively. Max position $= 2\,048$.

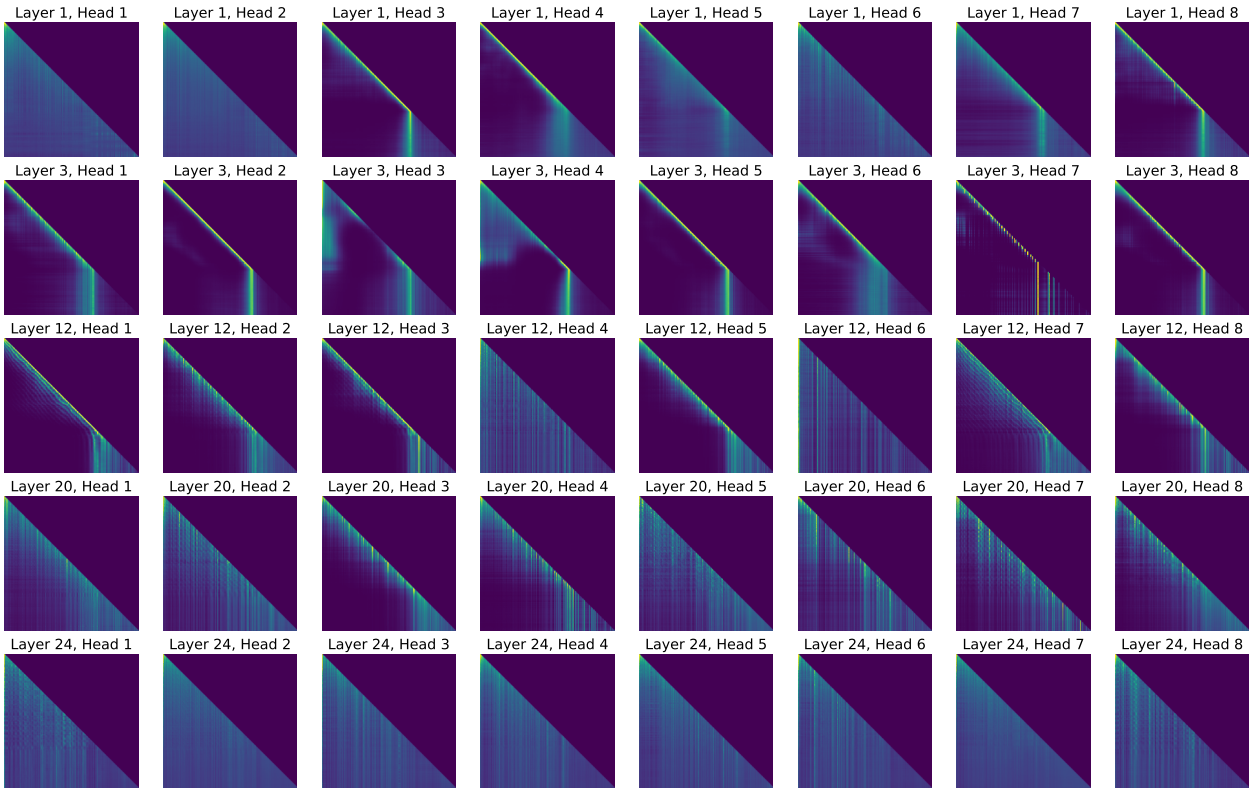


Figure 8. Full attention matrices of `APE` (baseline). X- and Y-axes are *key* and *query* positions respectively. Max position $= 3\,072$.

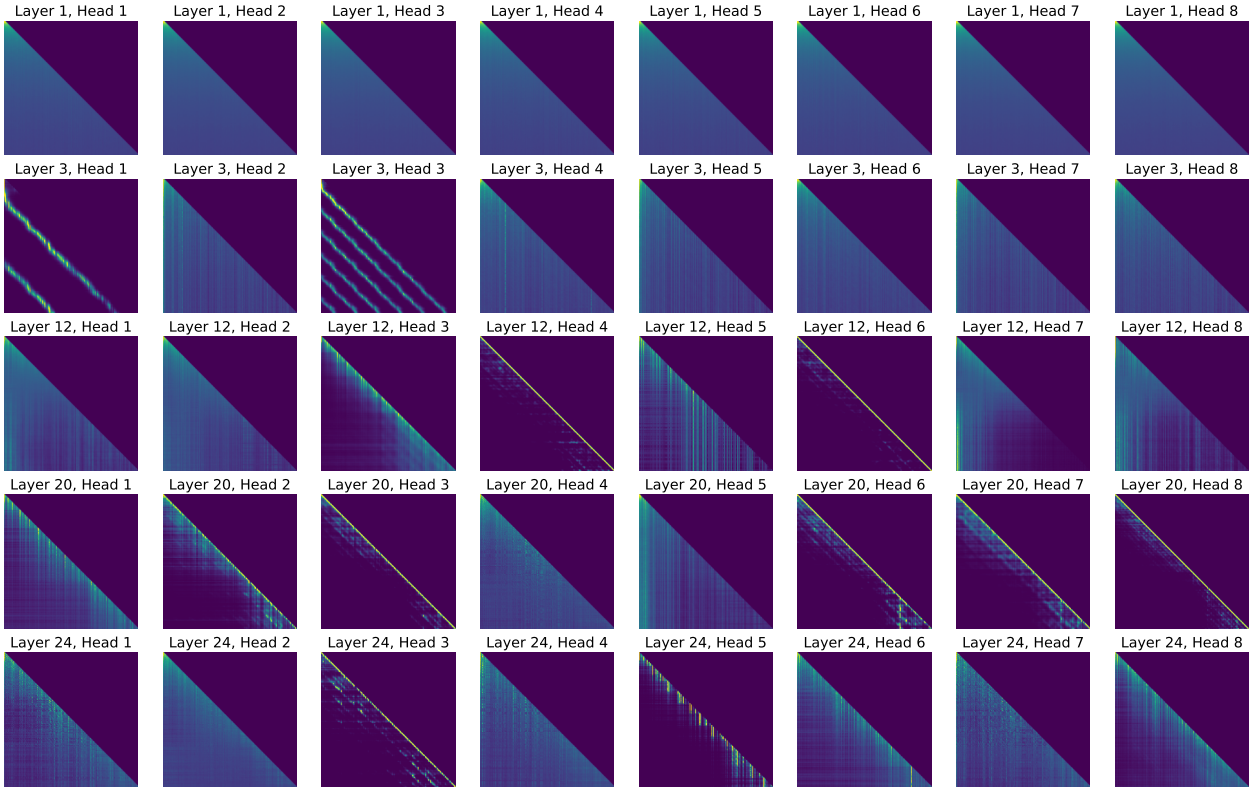


Figure 9. Full attention matrices of `sineSPE` (*with* gated SPE). Max token position $= 3\,072$.

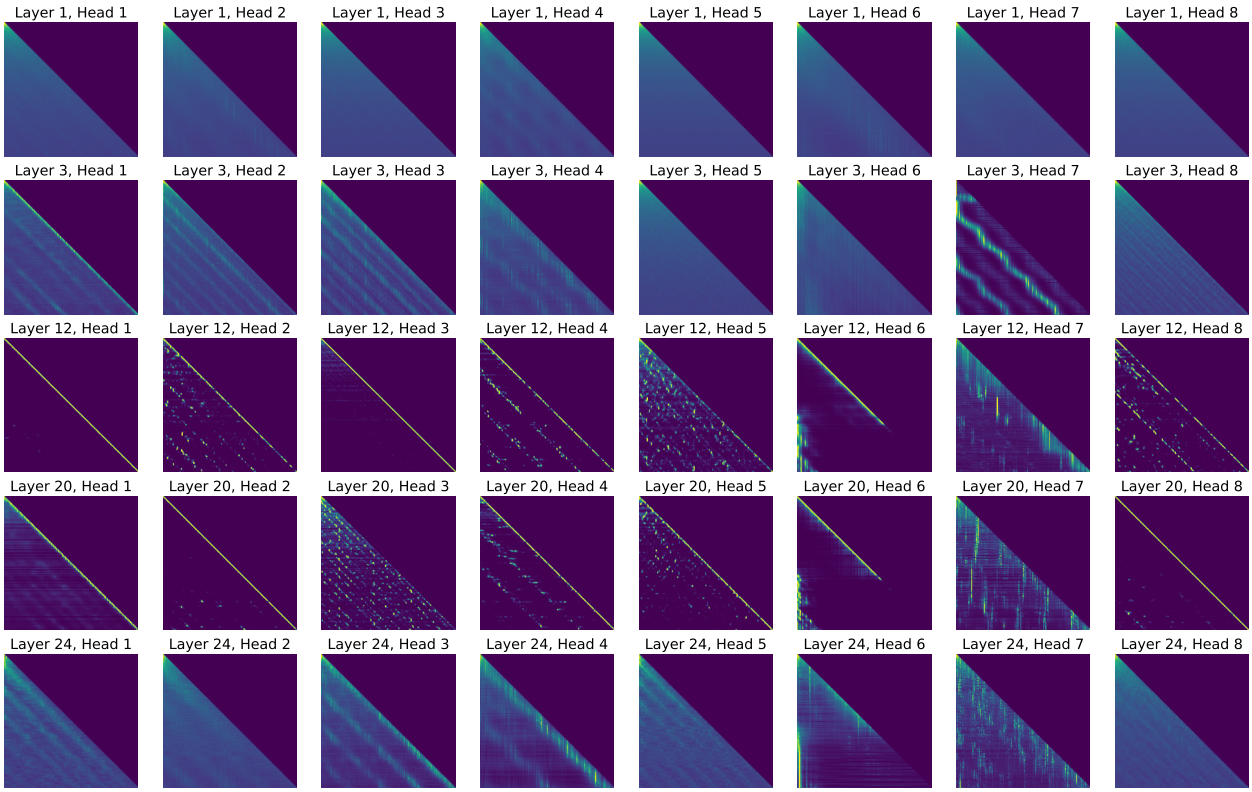


Figure 10. Full attention matrices of `sineSPE` (*without* SPE gating). Max token position $= 3\,072$.

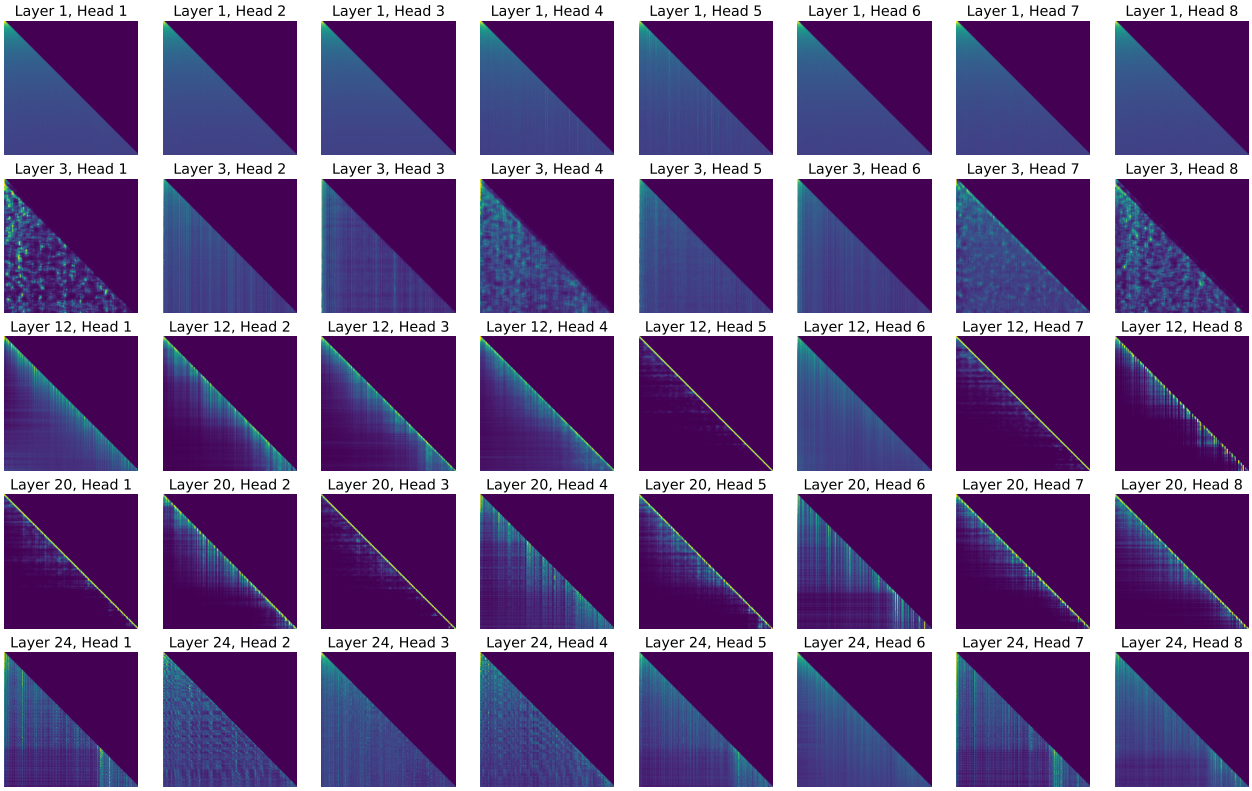


Figure 11. Full attention matrices of `convSPE` (*with* SPE gating, conv filter size $= 128$). Max token position $= 3\,072$.

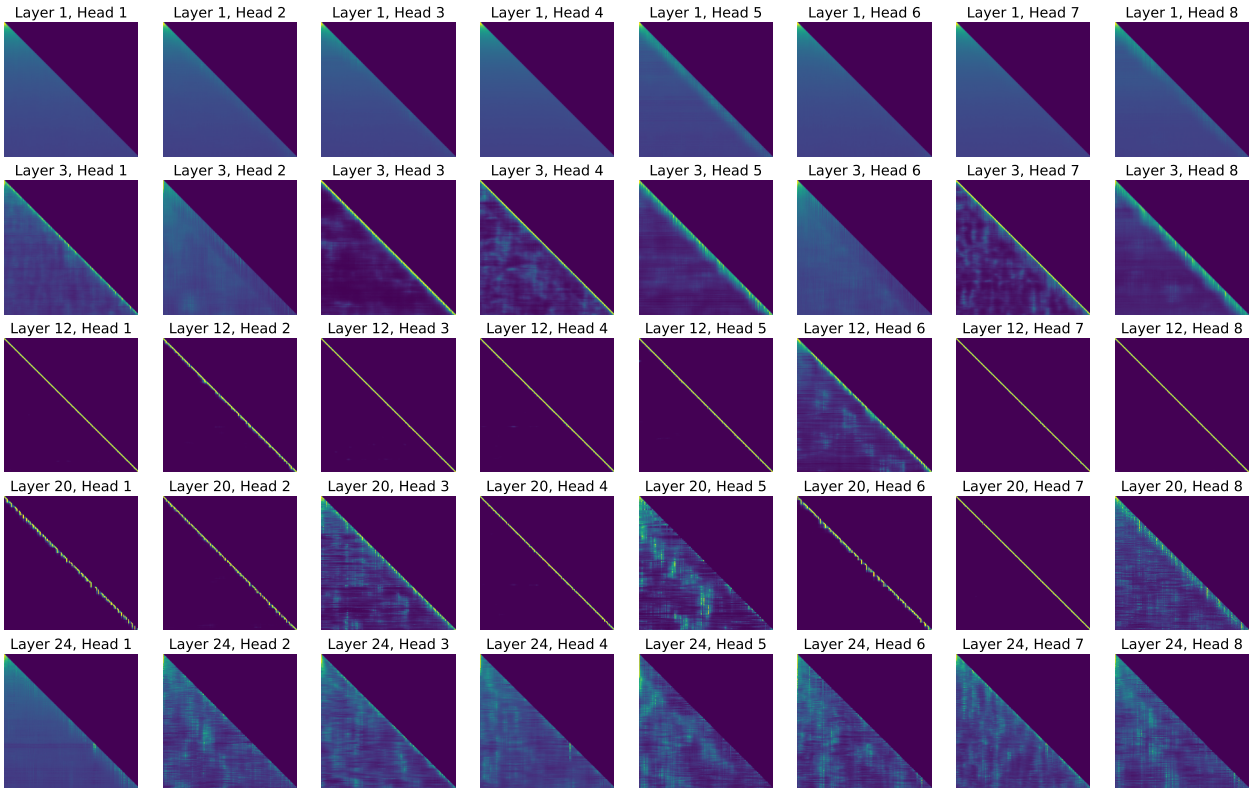


Figure 12. Full attention matrices of `convSPE` (*without* SPE gating, conv filter size $= 512$). Max token position $= 3\,072$.

tokens beyond position 2 048 (the training sequence length) seems to concentrate around 2 048 in earlier layers, rather than paying global or local attention. Such behavior is not seen in any of our SPE models. This potentially explains `APE`'s poor generalization to long sequences suggested by the stark increase in validation loss after position 2 048 (see Figure 3 in the main paper, and Table 6 here).

Next, comparing Figures 9 and 10, it is obvious that gated SPE gives the model the freedom to *switch off* PE in some heads to achieve global attention (see Figure 9), whereas the attention of ungated `sineSPE` (Figure 10) largely stays periodic, which might not be always desirable. The same can be said for `convSPE` (Figures 11 and 12). The gated `convSPE` is able to look much further back in the middle layers than its ungated counterpart.

D.2. Attention Visualization: CIFAR10

Figure 13 displays attention maps extracted from models trained on the LRA CIFAR10 task. Note that these are one-layer networks, and classification is done by prepending a special `CLS` token to the sequence of pixel values and using the output at this first position as input to a feed-forward classifier. Consequently, only the attention map at this single position (which is the one we display here) matters. (The model is therefore *de facto* not using self-attention, but rather attention with a single query and many keys. This removes the distinction between relative and absolute positions, which might explain why trainable APE performs better than SPE on this task.)

D.3. Evaluation of Desired PE Properties

We employ *identical word probing* and the associated metrics introduced in Wang et al. (2021) to compare the *translation invariance* and *monotonicity* properties of APE and our SPEs. The other properties mentioned in that work, namely *symmetry* and *direction balance*, are not evaluated here since the attention is uni-directional in our case. The models are also trained on pop piano music generation.

The metrics are calculated from attention matrices of each head in the 1st layer, averaged over all possible *identical-token* sequences (i.e., a sequence composed of repeated, same tokens; there are $\sim$340 of them for our REMI vocabulary). To eliminate the impact of applying row-wise softmax with causal masking on the translation invariance property, we compute the metrics on the *unnormalized* attention matrices, i.e., $\mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{D})$ for APE, and $\mathbf{A} = \exp(\widehat{\mathbf{Q}}\widehat{\mathbf{K}}^\top/\sqrt{R})$ for SPEs. Various combinations of *query positions* and *query-key offsets* are considered to examine whether the PE properties stay consistent when we extrapolate to longer sequences, as well as to look into their behavior in local and long-range attention spans.

We report the scores of the best-performing (i.e., lowest-scoring) head of each model in Table 7. From the table, we can notice that the PE properties of `APE` often deteriorate drastically in cases of extrapolation. On the contrary, the scores of ungated SPE models, i.e., models in which we enforce the incorporation of positional information in every layer, remain remarkably consistent throughout the positions. The evaluation here provides additional evidence for the extrapolation capability of SPEs.

D.4. Impact of the Number R of Realizations

In the main document, we discussed how SPE asymptotically leads to the desired cross-covariance structure as R grows to infinity. In this section, we empirically study how performance is affected by that parameter in practice. A first thing to highlight is that each training batch yields a new set of realizations for the noise $\mathbf{Z}_d$, so that the network sees the right attention pattern *on average*.

However, we may wonder whether how the number of realizations R impacts training and test performance. One can indeed notice that R may totally be set differently during training and inference, since it has no impact on the shape of the actual parameters/structure of the model. For this reason, we performed an ablation study where we use different values for R_{train} at training time, resulting in a trained model, and then evaluate its performance using a possibly different value R_{test}. The results are displayed in Figure 14.

We can notice that the result achieved with $R_{\text{test}} = R_{\text{train}}$ (highlighted in bold) is consistently close to the best result for the same R_{train}, and conversely, choosing $R_{\text{test}} \neq R_{\text{train}}$ often leads to a poor result. In other words, training and testing with the same R appears to be favorable for consistently good performance.

Another remarkable fact is that a higher R does not seem to imply better performance, even when $R_{\text{test}} = R_{\text{train}}$. On the contrary, `convSPE` achieved by far the highest accuracy with $R = 4$. This unexpected result seems contradictory to the fact that it means noisier attention patterns. Further investigation is required to explain this phenomenon, but we conjecture that this additional noise in the attention patterns leads to increased robustness of the trained model, helping generalization.

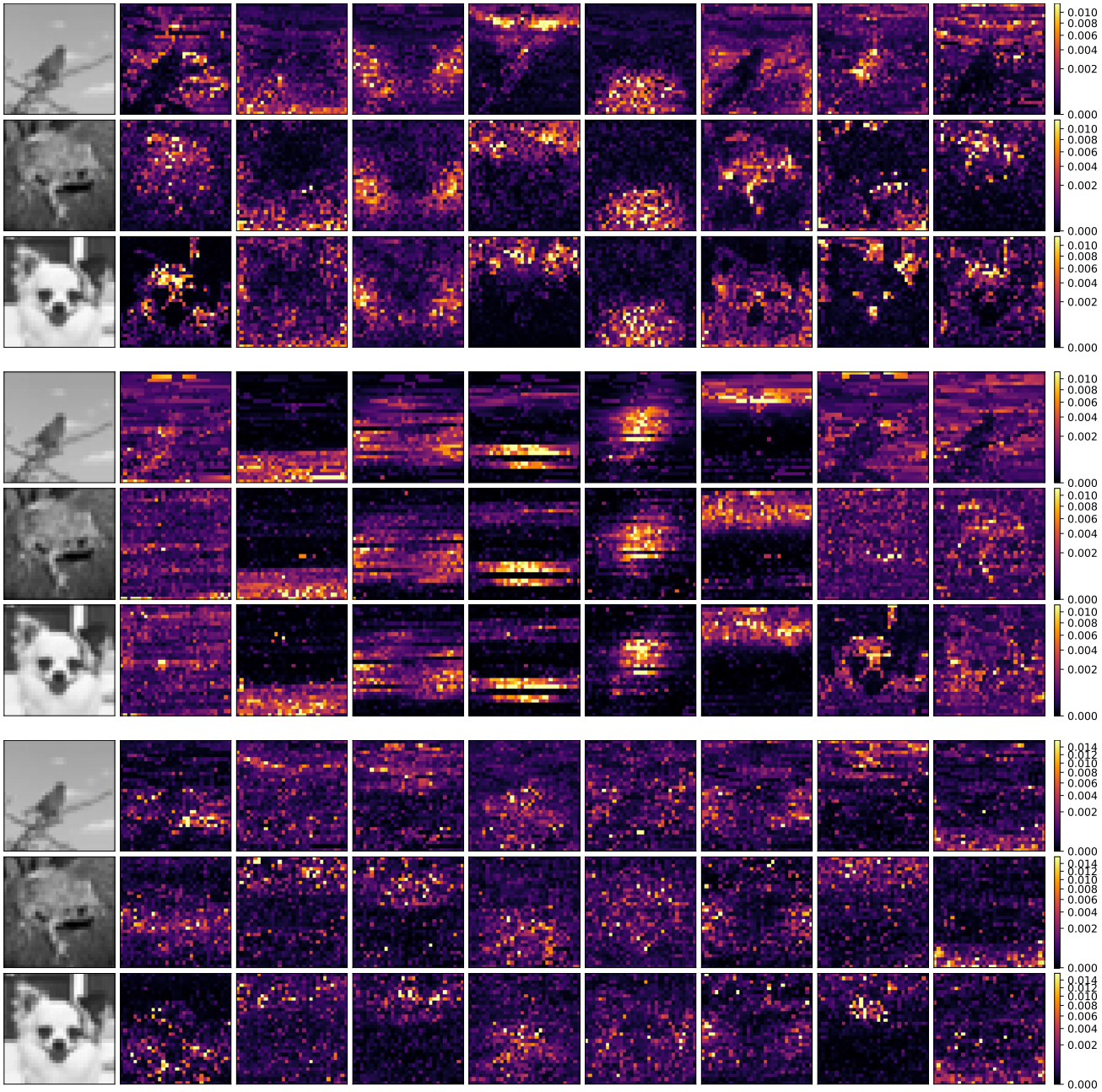
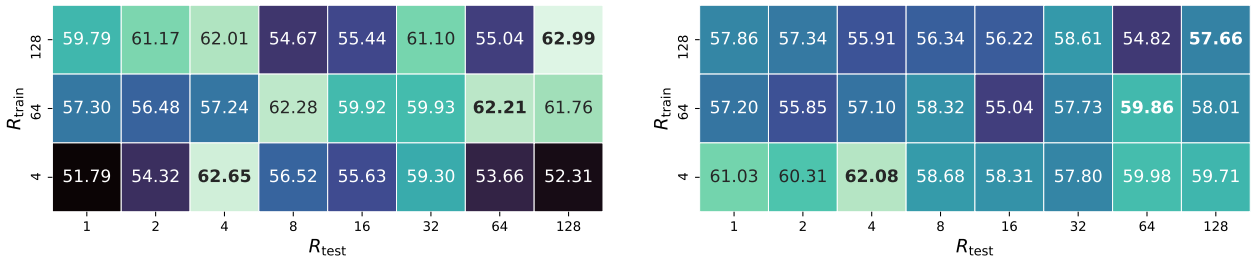


Figure 13. CIFAR10 attention maps for 3 variants of Linear Transformer-ReLU: learnable APE (top), `sineSPE` (middle), and `convSPE` (bottom). Each row displays the input image, followed by attention weights of the 8 respective heads for each pixel, with the special `CLS` token as the query.

Table 7. Evaluation of PEs metrics. `T:` translation invariance, `M:` monotonicity (lower is better). `ug`: models trained without SPE gating.

Query positions	$\mathbf{0 < pos \leq 1\,024}$			$\mathbf{1\,024 < pos \leq 2\,048}$			$\mathbf{2\,048 < pos \leq 2\,560}$ (extrapolation)		
Query-key offset	<128	<512	$<1\,024$	<128	<512	$<1\,024$	<128	<512	$<1\,024$
APE	T: 0.4335 M: **0.0152**	T: 0.2063 M: 0.0625	T: 0.1845 M: **0.0616**	T: 0.9142 M: **0.0193**	T: 0.6953 M: 0.0413	T: 0.6458 M: **0.0713**	T: 0.9599 M: 0.3974	T: 0.8959 M: 0.2429	T: 0.5886 M: 0.1637
sineSPE	T: 0.1660 M: 0.2893	T: 0.3078 M: 0.4406	T: 0.3527 M: 0.4283	T: 0.1337 M: 0.2826	T: 0.2504 M: 0.4063	T: 0.3228 M: 0.4167	T: 0.2167 M: 0.3253	T: 0.3599 M: 0.4060	T: 0.4147 M: 0.3913
sineSPE (ug)	T: **0.0141** M: 0.6295	T: 0.0242 M: 0.1844	T: 0.0231 M: 0.1582	T: **0.0135** M: 0.6238	T: 0.0206 M: 0.1623	T: 0.0190 M: 0.1061	T: **0.0105** M: 0.6189	T: 0.0196 M: 0.1609	T: 0.0163 M: 0.0994
convSPE	T: 0.3422 M: 0.1781	T: 0.5637 M: 0.2242	T: 0.6389 M: 0.2189	T: 0.3209 M: 0.1735	T: 0.6239 M: 0.3624	T: 0.7648 M: 0.4192	T: 0.3462 M: 0.1486	T: 0.6135 M: 0.3247	T: 0.7025 M: 0.2740
convSPE (ug)	T: 0.2828 M: 0.1234	T: **0.0192** M: **0.0249**	T: **0.0107** M: 0.0620	T: 0.3334 M: 0.1505	T: **0.0188** M: **0.0253**	T: **0.0109** M: 0.1254	T: 0.2207 M: **0.1342**	T: **0.0171** M: **0.0217**	T: **0.0106** M: **0.0989**



(a) `sineSPE` (b) `convSPE`

Figure 14. Accuracy of Performer-softmax with SPE on the LRA Text task, with different numbers of realizations R during training/testing. Each value is the result of a single run. Highlighted in bold are values obtained with $R_{\text{test}} = R_{\text{train}}$. Higher (brighter) is better.